제2권 통권9호 1989년 8월 1일 발행(매월 1일 발행) 1988년 10월 10일 등록(등록번호 라-3794) ㊏140-150 서울특별시 용산구 갈월동 17

퍼스널컴퓨터

하이테크

89/08

특집 / 한글 입출력에 관한 연구

유틸리티 / 계산기 프로그램 PANTA /
2 단편집이 가능한 더블에디터

테크닉 / 알라딘의 마술램프 /
프린터를 제어해 주는 PCNTL.COM

활용 / 프로텍트의 원리



□ 편집자 노트 □

이제는 황무지에 씨를 뿌려야 할 때

최근들어 세운상가나 용산전자상가에 나가보면 부쩍 사람들이 많이 붐비고 있는 것을 알 수 있습니다. 나이어린 중학생부터 회사원에 이르기까지 각계 각층의 사람들이 컴퓨터를 구입하기 위해 이곳 저곳 찾아다니며 상담을 하고 있습니다.

그런데 여기에 옛날과는 달리 한가지 변화가 일어나고 있다고 합니다. 옛날에는 대부분의 사람들이 컴퓨터를 구입하러 와서 성능이 좋은 것만을 찾았는데 최근들어서는 이것 저것 여러가지를 확인하고 각자의 용도에 맞는 기종을 선택한다고 합니다. 따라서 옛날처럼 적당히 소비자를 현혹시켜서 판매하기가 어렵다고 합니다. 그래서 이제는 그저 물건만 갖다 놓고 팔던 시대는 지난 것 같다고 말하는 주인도 있습니다.

컴퓨터에 대한 인식이 많이 달라졌다는 좋은 본보기가 아닌가 생각됩니다.

아무튼 이러한 여러가지 변화와 함께 올해중으로 보급될 PC의 댓수만도 25만대 가량이 될 것이라고 합니다. 상반기 중으로 일반판매가 약 7만대 가량 팔렸고 하반기 중으로는 그 2배 정도인 약 15만대 가량이 판매될 것이라고 각 업체에서 예상하고 있습니다. 그리고 교육용으로는 2~3만대 가량이 보급될 것이라고 합니다.

컴퓨터 1대에 10명씩만 계산해도 250만명이라는 숫자가 컴퓨터를 접하게 됩니다. 이전에 보급된 댓수까지 합치면 엄청난 사람이 컴퓨터를 접하고 있는셈입니다. 이렇게 많은 사람들이 컴퓨터와 함께 생활하고 있다고 말할 수 있습니다.

각종 매스컴에서 이제 본격적인 PC의 대중화시대에 접어들었다고 말하는 말들이 실감이 난다고 할 수 있습니다. 그러나 세계의 여러나라에 PC를 수출하고 주요 PC수출국으로 분류될 정도로 많이 생산하고 있는 현실과는 걸맞지 않게 모순된 점이 있습니다. 바로 소프트웨어의 빈곤입니다.

하드웨어는 많이 생산되고 있는데 비해 거기에 맞춰 개발된 우리의 소프트웨어는 전무하다시피 한 것이 우리의 현실입니다.

대기업들이 컴퓨터 하드웨어 생산에는 막대한 돈을 투자하여 키우는 데도 투자효과가 없다고 우리 실정에 맞는 프로그램의 개발에는 거의 투자하지 않아 떳떳하게 우리가 만든 프로그램이라고 내세울만한 것이 전혀 없습니다.

따라서 이제 막 컴퓨터를 배우려고 하는 어린 학생들을 대할 때면 부끄러워 얼굴을 들 수가 없습니다. 허울 좋게 말로만 PC수출대국이라고 떠들어대지만 가까운 이웃인 대만이나 일본 등과 비교해 볼 때면 우리의 열악한 소프트웨어 시장이 부끄럽기만 하고 무엇인가 크게 잘못되지 않았나 하는 생각마저 듭니다.

이제 본격적으로 PC교육이 시작되는 시점이므로 많은 학생들이 컴퓨터를 배우기 전에 이제부터 시작이라고 생각하고 더 이상 늦기 전에 우리의 토양에 맞는 소프트웨어의 씨를 뿌려야 할 때라고 생각됩니다. 그래야 그 씨가 뿌리내려 열매를 맺을 때 비로소 올바른 컴퓨터 문화가 싹튼다고 볼 수 있습니다.

우리 모두 황무지에 소프트웨어의 씨를 뿌리기 위해 다함께 노력합시다.

유용한 유틸리티 4
수퍼키

주희경/명지대학교 물리학과

이번에 소개할 컴퓨터 입문자를 위한 유틸리티 프로그램은 꽤나 특이한 것이다. 여태껏 컴퓨터를 배워오는 동안 키보드를 통하여 얼마나 많은 키가 입력되었을까 하고 생각해 본 적이 있는가? 이 말에 조금이라도 관심이 있는 독자는 지금하고 있는 작업에서 잠시 손을 떼고 그간 1년이든 2년이든, 아니면 단 몇개월이든 컴퓨터와 씨름(?)하면서 키보드의 키를 두드린 횟수를 생각해보기 바란다. 실로 엄청난 횟수임에 틀림없다. 그리고 이왕이면 그간 이렇게 많이 두드리고 있는 키의 기능들에 대해서 별다른 관심을 가져 보았는지에 대해서도 생각해 볼만 할 것이다. 또 홍수처럼 쏟아지는 각종 소프트웨어들을 생각해 볼때, 84키 또는 101키등으로 국한되어 있는 키보드내에서 과연 얼마나 많은 기능들을 창출해 낼 수 있을지에 대해서도 관심을 가져 볼 만한 것이다.

서두부터 필자가 이런 넋두리를 읊는 것은 다름이 아니라 키보드에 대해 흥미진진한 기능들을 발휘할 수 있는 수퍼키라는 유틸리티 프로그램을 소개하기 위해서이다.

수퍼키라는 이름에서도 쉽게 엿볼 수 있듯이 컴퓨터에 쓰이는 각종 키에 막강한 힘, 즉 다양한 기능들을 부여할 수 있는 프로그램이라는 것을 곧바로 직감할 수 있을 것으로 본다.

수퍼키에 대해서

원래 도구란 사용자가 그것을 어떻게 이용하는가에 따라 이용가치가 달라지게 된다. 올바르게 이용한다면 편리한 도구가 될 것이고 잘못 사용한다면 오히려 거추장스러운 도구가 될 것이다. 여기서 소개하려는 수퍼키는 미국 볼랜드사의 제품으로 디스켓 1장으로 구성되어 있으며 IBM-PC/XT/AT 또는 IBM 호환 기종에서 동작할 수 있다.

수퍼키 디스켓에는 다음과 같은 내용들이 있다.

- KEY.COM: 수퍼키 프로그램
- KEY.HLP: 수퍼키의 헬프메시지
- KEYINST.COM: 수퍼키의 인스톨 프로그램
- *.MAC: 각종 환경에서 사용할 수 있도록 정의되어 있는 매크로 화일.

수퍼키의 매력이라면 막강한 매크로 기능에 있다. 즉 일괄적으로 처리하는 반복 명령을 하나의 키에다 입축시키는 것이다. 그래서 언제든지 그 키가 눌려지면 지정된 모든 일을 순차적으로 실행한다.

예를 들어 파스칼 언어를 사용할 경우 프로시듀어나 while문과 같은 것은 일정한 틀을 가지므로 이것을 하나의 키를 이용하여 이 포맷을 출력시킨다면 그만큼의 시간을 줄일 수 있다. 이밖에도 화일을 다른 사람이 보기 힘들도록 감추어 주는 기능과 화면의 번-인(Burn-in) 현상(화면을 일정하게 켜놓으면 켜진 곳에 잔상이 남는 것)을 막기 위하여 일정 시간동안 키를 입력하지 않을 경우 화면을 자동적으로 지워주는 기능도 갖고 있다. (이 경우는 아무키나 누르면 화면이 살아난다).

이 지면에 수퍼키의 모든 기능을 다 싣는다는 것은 무리이다. 수퍼키의 기능들을 하나하나 서술해 가다보면 책 한권이 될 정도로 기능들이 방대하기 때문이다. 하지만 초보자를 위하여 흥미롭고 대략적인 키의 사용법을 위주로 전개해 나가려고 한다. 그러면 수퍼키를 사용하여 보자.

시작에 앞서

시작에 앞서 알아둘 사항이 몇가지 있다. 먼저 수퍼키는 램상주 프로그램이다. 그러므로 프로그램을 실행한 다음 뒤에 램상주 프로그램을 올리는 것은 삼가하기 바란다. 단, 가능한 프로그램은 사이드 킥 뿐이다. 이것은 나중에 수퍼키를 지운다든지 하는 경우 DOS가 제어할 수 없는 구멍이 생기기 때문이다. 이러한 경우 사용자는 컴퓨터를 다시 기동시켜야만 하는 불상사(?)가 발생한다. 수퍼키의 단점이라면 매뉴얼 없이는 사용이 불편하다는 것이다. 물론 모든 기능을 파악한다면 이야기는 달라지겠지만 말이다. 하여간 주의하여 사용하도록 한다.

램상주란 말그대로 프로그램을 디스크에서 주기억 메모리로 한번 로드시키면 계속 그곳에 남아있는 상태로 유지시켜 두는 것을 말한다. 그래서 다른 작업을 하다가도 특정 핫키를 누르면 이 램상주 프로그램이 동작하게 된다.

수퍼키의 구동

디렉토리를 보면 KEY.COM이 있는데 이것을 실행하면 **화면 1**과 같이 로고 화면이 나타난다. 만약 이와같은 현상이 일어나지 않는다면 KEYINST.COM을 실행하여 스크린 타입을 확인하여 보자. 왜냐하면 현재의 스크린 모드는 모노크롬으로 되어 있기 때문이다. 매뉴얼이 없던 필자도 이 부분에서 무척이나 헤맸다. 그런데 화면을 보면 로고만 나오고 키를 누를 때마다 클릭음을 내는 사실 외에 달라진 것이 없다.

화면 1이 나온 후부터는 언제든지 〈Alt〉+〈/〉키를 눌러보자. 그러면 화면 상단에 매크로, 코맨드, 펑션, 옵션, 디폴트, 암호화, 레이아웃, 셋업 등으로 구분되어 있는 메인 메뉴가 나타난다. 여기서부터 수퍼키의 활약이 시작된다. 앞으로 기술하면서 알아둘 것은 앞에서와 같이 '+'로 연결된 키의 조합, 〈Alt〉+〈/〉는 〈Alt〉키를 누른 상태에서 〈/〉키를 동시에 누르라는 것이다. 만약 수퍼키를 램상주에서 없애 버리려면 메인메뉴가 나타난 상태에서 〈Ctrl〉+〈Home〉+〈End〉키를 누른다. 그러면 다시 〈Alt〉+〈/〉키를 눌러도 메뉴는 나타나지 않을 것이다.

메뉴에 관하여

메뉴는 풀다운 방식을 사용하고 있기 때문에 화살표 키를 사용하여 메뉴를 선택하는 방법과 메뉴에서 대문자로 나타낸 키를 누름으로서 선택하는 방법이 있다. 메뉴의 모양이 마음에 들지 않는다면 수퍼키의 인스톨 프로그램인 KEYINST.COM을 실행시켜 화면의 종류와 메뉴의 패턴을 수정할 수 있다. 이 작업은 KEYINST.COM을 실행시켜 현재 사용하고 있는 화면의 종류(대부분은 허큘리스 카드를 사용하고 있는 흑백화면일 것이다.)와 메뉴에서 선택되어지는 항목들을 역상으로 표시하느냐 또는 밝게 표시하느냐 등의 여부만을 지정해 주기만 하면 된다.

다음은 이 상태(화면 맨 윗줄에 메인 메뉴가 나타나 있는 상태)에서 역상으로 되어 있는 부분을 화살표키로 움직여 'Macros'라는 메뉴를 선택해 보자. 또는 그대로 키보드에서 'M'자를 눌러도 된다. 그러면 **화면 2**와 같이 매크로 정의에 관한 서브 메뉴가 아래로 펼쳐진다.

나머지 메뉴들도 이와 마찬 가지 방법으로 선택하면 된다.

핫키의 사용

메인 메뉴를 거치지 않고 바로 원하는 명령을 실행할 수 있는 키를 핫키(Hot Key)라고 한다. 수퍼키에서도 이러한 것을 지원하는데 주요 핫키를 보면 **표 1**에 나타낸 바와 같다. 처음에 사용하는 사용자는 친절한 메뉴 선택이 손쉬울지 몰라도 좀 더 사용하다보면 거추장스러운 부분들을 많이 느낄 것이다. 그러므로 자주 사용되는 키는 어떤 특정 키에 의해 호출되게 하면 다소 편리하고 빠르게 작업을 할 수 있다.

이것이 핫키의 장점이다.

표 1의 첫번째 내용을 예로 살펴보면 매크로 정의의 시작을 선택하기 위해서 〈Alt〉+〈/〉키를 눌러 메인 메뉴를 만들고, 매크로정의 항목을 선택하기 위해 'M'을 누르고 다시 서브메뉴에서 정의시작을 나타내는 'B'를 선택하는 과정을 나타내고 있다. 이 모든 과정을 핫키로 사용하면 한순간에 해결할 수 있음을 보여주고 있다. 즉 〈Alt〉키를 누른 상태에서 〈=〉키만 누르면 된다.

〔화면 1〕 KEY.COM을 실행시켰을 때의 로고화면

```
Superkey                 Version 1.03A
                      IBM-PC/XT/AT/PCjr

Copyright (C) 1984,85 BORLAND Inc.


524288 bytes total memory
459136 bytes were free
402704 bytes free
  8000 bytes macro memory

B>
```

〔화면 2〕 매크로에 관한 서브 메뉴

```
Macros  Commands  Functions  Options  Defaults  Encryption  Layout  Setup
Begin
 Title
 Fixed Field
 Var Field
 autO-start
End
cUt and paste
eDit
Load
Merge
Clear
Save
```

터보 파스칼에서의 수퍼키

일반 프로그램을 작성하는 경우 우리는 같은 문법을 여러번 사용하여야 하는 경우가 생긴다. 가령 반복 루틴을 만드는 경우 for 루틴을 사용할 때 begin∽end를 사용하는데 만약 이러한 것을 매크로로 넣어두면 한번에 자동적으로 출력하게 된다. 파스칼에서 수퍼키를 사용하려면 먼저 수퍼키를 램상주시킨다. 즉 수퍼키를 실행시킨다. 다음은

〈Alt〉+〈/〉〈M〉〈L〉TURBO.MAC〈ENTER〉

와 같이 입력하여 매크로로 정의되어 있는 화일을 읽어들인다. 참고로 여기서 'L'은 메인 메뉴의 매크로 서브 메뉴에 있는 'Load', 즉 디스켓에 저장되어 있는 기존의 매크로 화일을 로드시키기 위한 것이다.

그 다음은 파스칼환경을 실행시킨다. 편의상 여기서는 볼랜드사의 터보 파스칼안의 내장 에디터를 기준으로 했다. 한가지 주의할 것은 새로운 매크로가 파스칼에서 이미 사용되는 핫키와 겹쳐서는 안되므로 이러한 점을 고려하여 잘 작성해야 한다. 여기서 사용된 TURBO.MAC은 수퍼키 디스켓에서 제공하는 매크로 화일인데 헬프화면 〈Ctrl〉+〈F1〉이 겹쳐지고 있었다. 여하튼 화면 3는 터보 파스칼에서 사용된 수퍼키의 헬프 화면을 보이고 있다.

동작을 확인하기 위해 에디터 상태에서 〈Alt〉+〈P〉를 눌러 보자. 그러면 화면상에 프로시듀어의 구조가 나오는데 프로시듀어명만 입력하면 모든 것을 자동적으로 만들어 주는 것을 확인할 수 있다. 나머지 구문도 마찬가지다. 매크로 기능을 알려면 〈Ctrl〉+〈F1〉를 누르면 된다.

〔표 1〕 수퍼키에 쓰이는 주요 핫키

명 령 어	핫 키	메뉴 방식 사용 키
Begin macro	〈Alt〉+〈=〉	〈Alt〉+〈/〉〈M〉〈B〉
Title	〈Alt〉+〈'〉	〈Alt〉+〈/〉〈M〉〈T〉
autO-start	〈Alt〉+〈`〉	〈Alt〉+〈/〉〈M〉〈O〉
End macro	〈Alt〉+〈-〉	〈Alt〉+〈/〉〈M〉〈E〉
Menu	〈Alt〉+〈/〉	〈Alt〉+〈/〉
Fixed field	〈Ctrl〉+〈[〉	〈Alt〉+〈/〉〈M〉〈F〉
Var field	〈Ctrl〉+〈-〉	〈Alt〉+〈/〉〈M〉〈O〉
cUt and paste	〈Alt〉+〈]〉	〈Alt〉+〈/〉〈M〉〈U〉
sKip macro	〈`〉	〈Alt〉+〈/〉〈C〉〈K〉
stoP	〈Ctrl〉+〈ESC〉	〈Alt〉+〈/〉〈C〉〈P〉
shoW titles	〈Alt〉+〈PrtSc〉	〈Alt〉+〈/〉〈C〉〈W〉
Command stack	〈Alt〉+〈\〉	〈Alt〉+〈/〉〈C〉〈W〉

〔화면 3〕 터보파스칼에서의 수퍼키 HELP 화면

```
                                                  ug     Break/watch
 This is your TURBO PASCAL macro help system      ndent    A:NONAME.PAS

  Press CtrlF1 anytime you want to see this help.
 Press Esc to close this window and return to TURBO

     AltT  invoke Turbo and include error messages
     AltW  invoke turbo for new Work file

           AltP  Procedure block
           AltF  Function  block
           AltC  Case Statement
           AltB  Begin / End pairs
           AltR  Repeat / Until Statement

           AltS Save and continue editing
           AltX save and eXit
           AltM save and then coMpile

------------------------------------------- Watch ------------------------

Alt: F1-Last help  F3-Pick  F6-Swap  F9-Compile  X-Exit
```

DOS상에서의 수퍼키

DOS상에서는 사용할 일들이 많다. 먼저 지나간 명령어를 다시 호출하는 방법을 알아보자. 화면에 메인 메뉴가 남아 있으면 〈ESC〉키를 눌러 지운다. 즉 이제 보통의 DOS 상태가 된 것이다. 여기서 'DIR'이든 'COPY' 명령이든 그외 어떤 것이라도 제한받지 않고 평상시에 DOS를 사용할 때처럼 명령들을 이용하고 〈Alt〉+〈\〉를 눌러 보자. 화면 4와 같이 화면 하단에 윈도우가 생기고 지금까지 입력했던 내용들이 순서대로 기억되어 있음을 볼 수 있을 것이다. 그리고 커서는 윈도우 안에 있게 되어 상하 화살표로 움직일 수 있는데 전에 입력했던 명령중에서 원하는 곳에 위치시킨 후 〈ENTER〉키를 누르면 그 기능이 수행된다.

참고로 도움말화면을 얻고 싶으면 〈F1〉을 누르면 된다(화면 5 참조).

유저정의 매크로화일

아마 웬만한 사용자라면 벌써 지루함을 느낄 것이다. 왜냐하면 기존에 있는 화일을 사용하는 것으로 만족하지 않기 때문이다. 앞서 잠깐 언급했듯이 수퍼키속에 있는 화일중에서 *.MAC로 되어 있는 것은 모두 특정 내용들을 담고 있는 매크로 정의 화일들이다. 앞에서는 이들 중의 하나인 TURBO.MAC를 이용한 것이다. 이제는 자신이 원하는 매크로 화일을 작성해 보기로 하자. 그리고 앞으로는 되도록이면 사용되는 키의 입력이 편하도록 핫키를 사용하도록 하겠다.

우선 할 일은 〈Alt〉+〈/〉〈M〉〈C〉라고 입력하여 매크로를 소거한다. 즉

〔화면 4〕 DOS에서의 수퍼키 이용

```
D:\SUPER>dir/w

 Volume in drive D has no label
 Directory of  D:\SUPER

.                   ..                  123       MAC     BASICA    MAC     COMMAND   COM
DOS       MAC       FOREIGN   MAC       FXFONT    EXE     GRAPH1    MAC     GRAPH2    MAC
IBM       DWN       IBM       FNT       KEY       COM     KEY       HLP     KEYINST   COM
LUNCH     MAC       MERGE1    MAC       MERGE2    MAC     MERGE3    MAC     PIVOT     FNT
SETFX     EXE       TAPE      MAC       TAPEOFF   MAC     TURBO     MAC     VOLKS     MAC
WORDPER   MAC       NEWDOS    MAC
       27 File(s)  10983424 bytes free

D:\SUPER> ========COMMAND STACK==========
          dir/w
          cls
          dir
          cls
          cd super
          cd..
          dir
          cls
          cd hardcopy
          PgUp/PgDn-page  ESC-exit=======
```

방금전까지 입력했던 명령들이 기억되어 있으며, 상하화살표키로 원하는 명령어에 위치시킨 후 〈ENTER〉 키를 치면 그 명령이 실행된다.

〔화면 5〕 DOS.MAC의 내용

```
F1: Help              F2: Sorted directory
F3: Disk status       F4: Disk cleanup
F5: Make directory F6: Change directory
F7: Remove - " -      F8: Copy file
F9: Delete file     F10: Backup
                                              OM        ATTRIB    EXE       HARDCOPY COM
    PgDn: more           Esc: End             OM        COMP      COM       DEBUG    COM
                                              OM        FDISK     COM       FIND     EXE
FORMAT   COM      GRAPHICS COM      HERC      COM       HGC       EXE       HIRES    COM
AUTOEXEC BAT      LABEL    COM      LINEPRNT  COM       CAT       COM       MEDRES   COM
MODE     COM      MONO     COM      MORE      COM       NKP       COM       INITPR   COM
CED      MAC      SPOOLER  COM      SIMCGA    COM       SORT      EXE       SYS      COM
TREE     COM      VDISK    SYS      CLRB      COM       HALT      COM       TED      COM
LX800    COM      CED      COM      CONFIG    SYS       CED       BAK       PRINTER  DEF
        45 File(s)       2048 bytes free
```

화면 1에서 'Clear'을 선택하면 기존에 로드된 매크로가 소거되는 것이다. 다음, 사용자가 원하는 키에 정의를 내린다. 만약 〈F10〉에 dir / w라는 기능을 갖추고 싶으면 〈Alt〉+〈=〉를 먼저 누른다. 이것은 앞에서 설명한 바대로 〈Alt〉+〈/〉를 눌러 메인 메뉴를 부르고 'Macros'와 매크로정의 시작을 선택하는 과정을 대신해 주게 된다. 다음은 원하는 키, 즉 〈F10〉를 누른 후 원하는 기능인 'dir / w'를 입력한다. 입력이 끝났으면 〈Alt〉+〈-〉를 누른다. 이것은 표 1에도 나와 있듯이 매크로 정의가 완료된 후 입력해야 하는 〈Alt〉+〈/〉 〈M〉 〈E〉의 과정을 대신해 주는 것이다.

만약 매크로에 관한 표제를 저장하고 싶다면 입력도중에 title을 입력하면 된다. 즉 핫키를 사용하지 않은 경우에는 〈Alt〉+〈/〉 〈M〉 〈B〉키를 눌러 매크로 키를 정의하고 다시 〈Alt〉+〈/〉 〈M〉 〈T〉를 누른다. 그러면 해당 키에 해당하는 메시지를 입력하라는 윈도우가 나타난다. 여기에 적절한 메시지를 입력한 후 〈Alt〉+〈/〉 〈M〉 〈E〉키를 눌러 빠져나오게 된다.

한편 핫키를 이용하는 경우는 위의 과정을 간단하게 다음과 같이 대체시킬 수 있다.

〈Alt〉+〈=〉 〈F10〉 dir/w 〈Enter〉 〈Alt〉+〈'〉 List of file 〈Enter〉 〈Alt〉+〈-〉

이같이 수퍼키에 숙련될수록 핫키를 사용하므로써 입력되는 과정들이 간단하게 될 수 있음을 알 수 있다.

만약 도중에 입력이 잘못되었을 경우에는 〈Ctrl〉+〈ESC〉키를 누르면 키 정의가 취소된다. 이렇게 하여 만들어진 타이틀은 〈Alt〉+〈/〉 〈C〉 〈W〉를 입력하면 지금까지 입력한 타이틀을 볼 수 있다. (여기서 'W'는 화면 1에 있는 서브메뉴의 'shoW titles'를 의미한다). 지금까지는 맛보기에 지나지 않는다. 그러면 수퍼키의 기능들을 조목조목 알아보기로 한다.

매크로의 일시 정지기능

매크로를 사용하다보면 그 키를 매크로가 아닌 본래의 키로 사용해야 할 경우가 생긴다. 이러한 경우 〈Alt〉+〈'〉키를 누르면 한번만 그 키의 매크로 기능이 일시 정지한다.

핫키를 사용하지 않는 경우는 〈Alt〉+〈/〉로 메인메뉴를 화면에 나타나게 한 다음 코맨드(C)를 선택한 후 서브메뉴의 스킵매크로(sKip macro)를 선택하면 된다. 다음 순서로 입력을 해보자.

〈Alt〉+〈/〉〈M〉〈B〉: 매크로 정의를 시작한다.

〈F3〉: 〈F3〉키를 재정의하고자 한다.

dir/w: DOS에서의 템플리트 기능을 갖고 있는 〈F3〉키를 'dir / w'란 명령으로 재지정하고자 한다.

〈Alt〉+〈/〉〈M〉〈E〉: 〈F3〉키의 매크로 정의를 끝마친다.

〈F3〉: 〈F3〉키를 누른다. 그러면 본래의 템플리트 기능을 상실하고 DOS의 'dir / w'명령이 수행된다.

이제 이 〈F3〉키를 본래의 템플리트용으로 쓰고자 한다.

cls: DOS 프롬프트상에서 'cls'라고 입력한다.

〈Alt〉+〈/〉〈C〉〈K〉: 메인메뉴의 코맨드 서브메뉴에서 스킵을 선택한다.

〈ESC〉: 〈ESC〉키를 눌러 보통의 DOS 상태로 돌아온다.

〈F3〉:〈F3〉키를 누르면 'cls'명령이 실행된다(한번만 실행됨).

매크로의 읽기와 저장

수퍼키에서 정의한 키 매크로들은 KEY.COM이 메모리에 상주되어 있는 동안에만 동작되기 때문에 특별히 자주 이용되는 매크로들은 디스크에 저장해 두면 편리하게 사용할 수 있다.

〈Alt〉+〈/〉〈M〉〈S〉

와 같이 입력하여 'Save'메뉴를 선택한다. 그 다음은 원하는 화일이름을 기입하면 된다. 여기서 다시 읽어들일 경우는 같은 메뉴(Macro)에서 'Load'를 선택하여 화일이름을 입력해주면 된다.

로드시 자동 실행 매크로

어떤 매크로는 불러들일때 자동적으로 실행할 수 있도록 하면 편리하다. 여기서는 그 예로 DOS 프롬프트 세트에 관하여 자동 실행 매크로를 만들어 보기로 하자.

먼저 〈Ctrl〉+〈F10〉키에 이와 같은 기능의 매크로를 정의하자.

〈Alt〉+〈=〉〈Ctrl〉+〈F10〉 PROMPT $P $T $G 〈ENTER〉

이것을 자동실행 매크로로 정의하려면

〈Alt〉+〈/〉〈M〉〈O〉

키를 눌러 'autO-start' 메뉴를 선택한다. 다음은 보통 매크로와 같이 끝낸다.

〈Alt〉+〈-〉 (핫키 이용)

이제 이것을 화일로 저장한 후 다시 읽어들이면 〈Ctrl〉+〈F10〉이 눌려진 것과 같은 기능을 실행하게 된다.

도움말 윈도우의 작성

화면상에 임의의 키에 대한 도움말

〔화면 6〕 사용자가 만든 전화번호 화면

```
B>dir/w

 Volume in drive B has no label
 Directory of  B:\

123      MAC    BASICA   MAC    COMMAND  COM    DOS      MAC    FOREIGN  MAC
FXFONT   EXE    GRAPH1   MAC    GRAPH2   MAC    IBM      DWN    IBM      FNT
KEY      COM    KEY      HLP    KEYINST  COM    LUNCH    MAC    MERGE1   MAC
MERGE2   MAC    MERGE3   MAC                                    TAPE     MAC
TAPEOFF  MAC    TURBO    MAC                                    NEWDOS   MAC
TEL      BAK    TEL      MAC     Kitty              857-6952
       27 File(s)      92160 b   DDolbee            373-4521
                                 Gargamel           719-5678
B>                               Maro               386-8276
                                 Ahnjb              843-5325
                                 EMPAL-BBS          967-8022
                                 KETEL              312-2868
                                 DACOM-NET          742-0201
```

을 만들 수 있다. 이것은 수퍼키의 자랑할 만한 기능중의 하나이다. 사용자가 메모를 하고 이것을 항상 보아야 한다면 수퍼키가 이런 일을 대치할 수 있다. 이 윈도우를 작성하려면 적당한 스크린 에디터를 준비하여 그곳에 다음과 같이 텍스트 화일을 작성해 보자.

```
〈BEGDISP〉 〈AltF10〉 30, 10, 30,10

  Kitty          857-6952
  DDolbee        373-4521
  Gargamel       719-5678
  Maro           387-8276
  Ahnjb          843-5325
  EMPAL-BBS      967-8022
  KETEL          312-2868
  DACOM-NET      742-0201

〈ENDDEF〉
```

위와 같이 작성된 화일을 만들어 놓고 수퍼키의 'Load'를 선택하여 읽어들인다. 이제는 〈Alt〉+〈F10〉을 누르면 **화면 6**과 같이 화면의 중간에 나타나게 된다. 위 그림의 설명은 다음과 같다.

〈BEGDISP〉는 윈도우의 시작을 말한다.
〈AltF10〉은 사용자가 정하는 임의의 키이다.
30, 10, 30, 10. 시작 컬럼, 시작 라인, 가로 길이, 세로 길이.
화면에 들어갈 내용은 크기에 제한이 없다. 만약 지정한 행보다 많아지면 자동적으로 스크롤된다.
〈Ctrl〉+〈L〉 화면작성 도중 다음 페이지로 넘길때 사용한다.
〈Ctrl〉+〈B〉 텍스트의 색상을 변환시킨다.
〈Ctrl〉+〈D〉 프레임 색을 조절한다.
〈ENDDEF〉 매크로 정의가 끝났음을 알린다.

매크로 실행중의 키입력을 가변 또는 고정 필드라고 하는데 매크로를 실행하는 과정중에 키보드로 입력을 받아야하는 경우가 종종 생긴다. 다음의 예를 보자. 특정 디렉토리의 특정 화일들을 다른 곳으로 복사하고, 원래의 화일은 지우려고 한다면 명령을 여러번 입력하여야 한다. 이것을 해결하려면 다음과 같이 매크로를 만든다.

〈Alt〉+〈=〉〈Alt〉+〈C〉
COPY 〈Space〉 〈Alt〉+〈/〉〈M〉〈V〉〈Alt〉+〈/〉〈M〉〈V〉〈Space〉 A:〈Enter〉
〈Alt〉+〈-〉

이것은 키로 입력받은 화일만을 복사하는 매크로이다. 그러나 우리가 원하는 것은 이것이 아니다. 키보드로 입력을 받아서 복사를 한 후에 그 곳에 있는 화일을 제거하는 것이다. 그러면 어떻게 할 것인가. 다음의 방법을 보자.

〈Alt〉+〈=〉 〈Alt〉+〈C〉

〔화면 7〕 각 메뉴에서 펼쳐지는 서브메뉴

```
Macros   Commands  Functions   Options   Defaults  Encryption Layout  Setup

Begin
 Title
 Fixed Field
 Var Field
 autO-start
End
cUt and paste
eDit
Load
Merge
Clear
Save

Commands  Functions   Options   Defaults  Encryption Layout  Setup

Command stack
sKip macro
stoP
Sidekick
shoW titles

Options    Defaults  Encryption Layout  Setup

Arrow keys          OFF
Bottom line         OFF
Command stack       ON
Format fields       OFF
Keyb. click         ON
One finger          OFF
Playback delay        0
proTect delay         3
sUspend             OFF
disk Wait           OFF
Save options

Defaults  Encryption Layout  Setup

Begin macro      AltHOME
Title            Alt'
autO-start       Alt'
End macro        AltEND
Menu             Alt/
Fixed field      Ctrl]
Var field        Ctrl-
cUt and paste    Alt]
beGin block      B
sKip macro       '
stoP             CtrlESC
shoW titles      AltPRT
Command stack    Alt\
Decimal point    .
deLimiter        ,
fill cHaracter   <SPACE>
help sYstem      F1
Arrow keys
Save defaults
```

COPY 〈Space〉〈Alt〉+〈=〉〈Ctrl〉+〈F10〉
〈Alt〉+〈←〉〈Space〉 A: 〈Enter〉
ERASE 〈Space〉〈Ctrl〉+〈F10〉 〈Enter〉

〈Alt〉+〈←〉

위의 경우는 임의의 선택된 〈Ctrl〉+〈F10〉에 키보드 입력받은 것을 잠시 보관한다. 그런 다음 다음번 ERASE 명령이 나올때 다시 사용하는 것이다. 즉 〈Ctrl〉+〈F10〉은 변수로 작용한 것이다.

화면 숨기기 기능

업무중에 아니면 오락을 하던 중에 잠시 나갔다 오는 경우가 있다. 그러면 누군가가 컴퓨터의 키보드를 만지는 것을 종종 본다. 이러한 것을 방지하기 위해서 수퍼키에서는 특정키가 눌려지기 전까지 화면을 감추는 기능이 있다.

우선 다음과 같이 따라서 지정한다.

〈Alt〉+〈=〉〈Ctrl〉+〈0〉 〈Alt〉+〈/〉
〈F〉〈S〉〈Alt〉+〈←〉
〈Alt〉+〈=〉〈Ctrl〉+〈0〉 〈Alt〉+〈/〉
〈F〉〈0〉〈Alt〉+〈←〉

여기서 〈Ctrl〉+〈0〉과 〈Ctrl〉+〈1〉은 사용자가 임의로 지정할 수 있다. 이제 〈Ctrl〉+〈0〉을 누르면 화면에는 아무것도 없게 된다. 다시 〈Ctrl〉+〈1〉을 누르면 상태로 돌아간다.

임포트(Import) 기능

임포트 기능이란 사이드 킥에도 있는 기능인데 실행중인 화면의 특정부분을 원하는 곳에 위치시키는 것을 말한다. 수퍼키에서는 이것을 매크로에 입력하여 여러번 반복할 수 있다는 것이 사이드 킥과는 다르다. 오히려 여러번 반복할 수 있어 효율적일 수 있다. 사용법은 다음과 같다. 실행중인 화면에서 복사할 것이 있다면

〈Alt〉+〈]〉 〈Ctrl〉+〈1〉

을 입력하여 〈Ctrl〉+〈1〉에 읽혀질 것을 알려준다. 다음 원하는 시작 위치에 커서를 가져간 후 〈B〉키를 누른다. 다음 커서를 움직이면 화면이 반전되는데 이것은 입력받을 부분을 나타낸다. 원하는 부분이 정해졌으면 〈Enter〉키를 누른다. 이로써 〈Ctrl〉+〈1〉키에는 화면의 원하는 부분이 정의되었다.

맺음말

수퍼키에는 그밖에도 시간을 나타내주는 기능, 화면을 숨기는 기능 또는 소리조절기능등 여기에 소개한 것보다 그렇지 못한 기능들이 더 많다고 할 수 있다. 하지만 이 많은 것을 기사화 한다는 것은 무리이고 해서 주요부분만을 서술하려 했는데 흐름이 너무 산만하지 않을까 하는 염려가 된다.

각 메뉴에 해당하는 서브메뉴들을 화면 7에 실었으니 참고하기 바란다.

바르게 알고 사용합시다 1
화일속성

김대순 / 아주대 아미콤

성자가 된 청소부의 이야기중에는 자반이라는 청소부와 그 어머니의 돼지를 이용한 청소이야기가 있다. 그 내용을 잠깐 들여다 보자.

자반은 하루종일 쉬지않고 일을 했고, 사람들은 그 댓가로 집집마다 매달 일정한 수고비를 그에게 지불했으며, 매일같이 음식도 주었다. 자반이 부양할 가족이라고는 어머니 한 분 뿐이었으므로 음식이 넉넉하였다. 자반의 어머니는 자반이 얻어오는 음식중에서 먹고 남은 것으로 돼지를 치기 시작했고, 돼지들은 거리를 청소하는데에 큰 도움이 되었다.

아침 일찍 자반의 어머니가 문을 열어주면 돼지들은 거리로 달려 나가 잔뜩 쌓인 쓰레기들을 먹어 치웠으므로 할일이 그만큼 덜어졌다. 이따금 하수구가 막히면 돼지들이 하수구 속으로 들어가 구멍을 막고 있는 쓰레기 더미를 밀어내 주기도 했다.

효과적인 화일의 소트

하드디스크를 여러가지 업무에 맞게 사용하다보면, 루트디렉토리에는 다수의 화일과 디렉토리들이 서로 섞여 있어 지저분하게 된다. 이런 경우 보기에도 좋지 않고, 디스크의 효율적 관리라는 면에서도 불합리한 디스크의 관리라는 지적을 받게 된다. 따라서 하드디스크를 각 기능별로 서브디렉토리들을 구성하여 각 화일들을 그 기능에 또는 사용자에게 설정된 서브디렉토리로 저장하면 편리하다.

이렇게 화일들을 서브디렉토리로 옮길때 루트디렉토리에 있어야 할 COMMAND. COM이나 CONFIG.SYS, 또는 AUTOEXEC.BAT화일을 서브디렉토리로 옮기는 경우가 있다.

따라서 루트디렉토리에 이러한 화일들이 없을 경우 시스템은 부트시 필요한 COMMAND.COM과 각종 디바이스를 설치하기 위한 CONFIG.SYS화일을 루트디렉토리에서 찾기 때문에 부트에 실패하게 된다.

하지만 RAMDRIVE.SYS와 MOUSE. SYS 같은 화일들은 화면 1과 같이 CONFIG. SYS에 그 패스를 지정할 수 있어 서브디렉토리에 저장해도 상관없다.

그러나 루트디렉토리에 필요한 이 화일들을 그대로 둘 경우 서로 뒤섞여 있어 미관상 매우 좋지 않다. 그러면 이를 해결할 방법은 없을까?

여기서 성자가 된 청소부 자반의 돼지를 이용한 효율적인 청소 방법의 지혜를 빌려 하드디스크를 정리해보자. 우리가 이용할 돼지(?)는 노턴 유틸리티의 DS.EXE와 FA.EXE, 그리고 도스 유틸리티의 ATTRIB.EXE, PC MAGAZINE의 ATTR.COM 또는

〔화면 1〕 CONFIG.SYS에 패스 지정한 예

```
C:\>TYPE CONFIG.SYS
FILES = 20
BUFFERS = 15
DEVICE = C:\SYS\DMDRIV.BIN
DEVICE = C:\SYS\EMS286.SYS 320
DEVICE = C:\SYS\MOUSE.SYS

C:\>
```

〔화면 2〕 디렉토리들을 소트한 예

```
C:\>DS E > NULL

C:\>DIR

 Volume in drive C is DISK1_VOL1
 Directory of  C:\

UTILITY      <DIR>      7-13-89  11:38a
PASCAL       <DIR>      7-13-89  11:38a
PROLOG       <DIR>      7-13-89  11:38a
WP           <DIR>      7-13-89  11:38a
KHAN         <DIR>      7-13-89  11:38a
MDS          <DIR>      7-13-89  11:39a
SKPLUS       <DIR>      7-13-89  11:39a
ORCAD3       <DIR>      7-13-89  11:39a
COMMU        <DIR>      7-13-89  11:39a
DBASE        <DIR>      7-13-89  11:39a
TD           <DIR>      7-13-89  11:39a
DRHALO       <DIR>      7-13-89  12:24p
FORTRAN      <DIR>      7-13-89   1:23p
WORK         <DIR>      7-13-89   1:31p
TC           <DIR>      7-14-89   9:37a
AUTOEXEC BAK       214  7-13-89   8:44p
AUTOEXEC BAT       207  7-13-89   8:49p
VKETTE   COM      5120  6-01-86   8:00a
CACHE-AT COM      6656  8-01-86   8:00a
COMMAND  COM     25307  5-31-89  12:00p
DISKPARK COM      4136 10-13-88  12:00p
VSCREEN  COM      2560  8-01-86   8:00a
ONLINE   HLP     29219 10-13-88  12:00p
CONFIG   SYS        93  7-13-89   4:21p
       24 File(s)  12052480 bytes free

C:\>
```

PCSHELL.EXE이다.

제일 먼저 돼지 DS.EXE를 이용하여 화일을 확장자 순으로 루트디렉토리를 **화면 2**와 같이 소트해 보자. 돼지 DS.EXE의 사용법은 아래와 같다.

DS[소트할 디렉토리의 이름]

또는

DS 소트 기준[소트할 디렉토리의 이름] [/S]

소트기준 N-화일순으로 기정
E-화일 확장자순으로 지정
D-날짜순으로 지정
T-시간순으로 지정
S-화일 크기순으로 지정
A-소트를 역순으로 할 경우
/S는 서브 디렉토리도 소트할 경우 지정한다.

또는 **화면 3**과 같의 돼지 PCSHELL.EXE의 디렉토리 소트 기능을 이용하여 소트 기준을 지정한 다음 소트를 하면 된다.

화면 2에서 보듯이 디렉토리들을 어느 정도 구분하고, 디렉토리와 화일확장자순으로 소트했음에도 꼭 어디가 막힌(?)것 처럼 아직 깨끗하게 보이지 않는다. 그러면 이번에는 **화면 4**에서 보는 것처럼 PCSHELL의 화일기능 항목의 화일속성변경 기능을 이용해 보자.

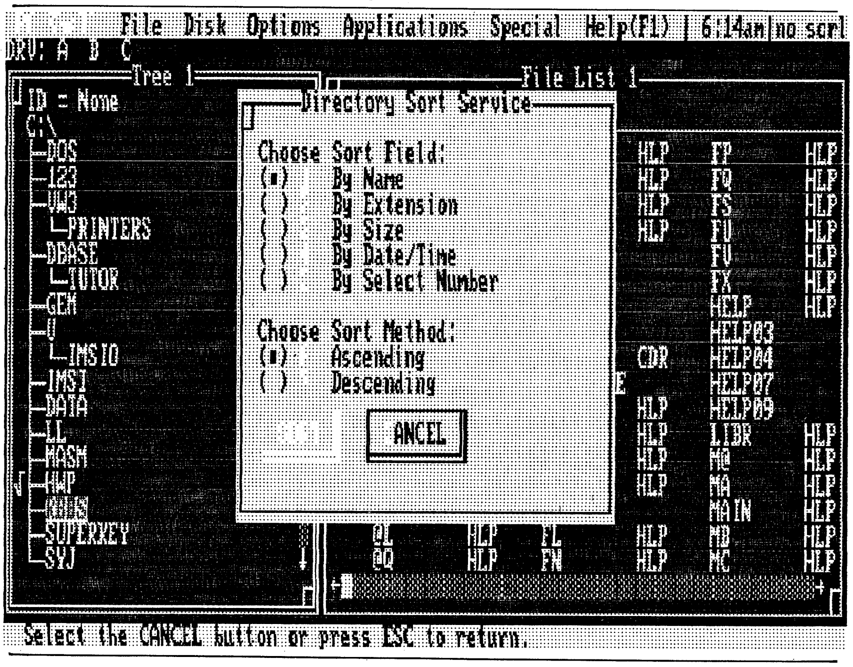
먼저 PCSHELL을 기동시킨 다음, 고치고자 하는 화일을 리턴키로 선택한다. 그런 다음 〈ALT〉키를 눌러 기능 항목으로 제어를 옮긴 후, F키를 눌러 화일기능을 선택한다.

여기서 A키를 누르면 화일속성변경 기능을 행하게 되는데, 화면에서 보는 것과 같이 H, S, R, A라는 네가지의 화일속성을 그 해당 문자를 누름으로써 선택한다. H키는 DIR 도스 명령에서 제외시키는, 감추는 화일 속성을 지정하기 위한 것이고, S키는 시스템 화일속성을 설정한다.

R키는 화일을 읽기만 할 뿐 쓰거나 지우지 못하게 하기 위한 속성이다. 또한 A키는 화일을 읽고/쓰기가 가능하게 한다. 이렇게해서 속성을 변경하고자 하는 화일들에 대한 모든 처리가 끝났을 경우 U키를 눌러 수정한 내용을 디스크에 수록하여 디스크의 화일속성을 완전히 변경할 수 있다.

PCSHELL 이외에 화일속성을 변경하는 유틸리티로는 도스에서 제공하는 ATTRIB.EXE와 노턴의 FA.EXE, 그리고 PC MAGAZINE의 ATTR.COM 등이 있다. 이 유틸리티들은 PCSHELL과 같이 디렉토리들의 사이를 간단한 키조작으로 자유스럽게 이동하여 많은 화일들을 제어할 수 있고, 도스 명령어 수준에서만 조작이 가능하기 때문에

〔화면 3〕 PCSHELL의 화일속성 변경



〔그림 1〕 디렉토리 엔트리 구성

0H	08H	0BH	0CH	10H	16H	18H	1AH	1CH
화일명	확장자	속성	예약영역	예약영역	변경시각	변경 날짜	클러스터	화일크기

〔그림 2〕 속성 바이트의 의미

비트 5	비트 4	비트 3	비트 2	비트 1	비트 0
보존	디렉토리	볼륨레이블	시스템 화일	감추어진 화일	읽기만 하는 화일

화일에 대해 일반적인 화일 쓰기가 행해질 때 1로 된다.

디렉토리의 이름을 뜻한다.

디스켓을 포맷할 때, 11자의 볼륨레이블을 받을 때 1로 된다.

이것은 화일이 시스템화일인 것으로 사용되어 사용자로부터 보호를 받기 위한 것이다. 이 비트가 1로 되면 일반적인 도스기능 호출로는 이 화일을 액세스하지 못한다.

이 화일이 히든 화일인지를 설정하는 것으로 일반적인 탐색에서 제외되어 COPY가 되지 않지만 액세스는 가능하다.

1로 세트되면 읽기만 하고, 쓰기는 못한다.

〔화면 4〕 디렉토리 영역의 예

```
C:\DOS>DEBUG
-L DS:0 1 5 1      드라이브 B의 디스크의 디렉토리 영역을 읽는다.
-D DS:0 L100       읽은 디렉토리 영역을 본다.
1176:0000  44 45 42 55 47 20 20 20-42 41 4B 20 00 00 00 00   DEBUG   BAK ....
1176:0010  00 00 00 00 00 00 85 09-21 00 02 00 00 55 00 00   ........!....U..
1176:0020  44 45 42 55 47 20 20 20-47 45 4D 20 00 00 00 00   DEBUG   GEM ....
1176:0030  00 00 00 00 00 00 E6 09-21 00 15 00 00 53 00 00   ........!....S..
1176:0040  32 20 20 20 20 20 20 20-44 42 47 20 00 00 00 00   2       DBG ....
1176:0050  00 00 00 00 00 00 79 02-21 00 04 00 99 01 00 00   ......y.!.......
1176:0060  31 20 20 20 20 20 20 20-44 42 47 20 00 00 00 00   1       DBG ....
1176:0070  00 00 00 00 00 00 16 02-21 00 05 00 F3 04 00 00   ........!.......
1176:0080  48 57 50 20 20 20 20 20-43 46 47 20 00 00 00 00   HWP     CFG ....
1176:0090  00 00 00 00 00 00 2F 09-21 00 07 00 1E 04 00 00   ....../.!.......
1176:00A0  46 49 4C 45 41 54 54 52-48 57 50 20 00 00 00 00   FILEATTRHWP ....
1176:00B0  00 00 00 00 00 00 5D 0A-21 00 5C 00 12 B8 00 00   ......].!.\.....
1176:00C0  53 43 52 20 20 20 20 20-20 20 20 20 00 00 00 00   SCR         ....
1176:00D0  00 00 00 00 00 00 82 08-21 00 78 00 80 01 00 00   ........!.x.....
1176:00E0  46 41 20 20 20 20 20 20-48 57 50 20 00 00 00 00   FA      HWP ....
1176:00F0  00 00 00 00 00 00 72 0A-21 00 79 (00) 12 B8 00 00   ......r.!.y.....  → 이것이 속성 바이트이다.
1176:0100  46 49 4C 45 41 54 54 52-42 41 4B 20 00 00 00 00   FILEATTRBAK ....
1176:0110  00 00 00 00 00 00 F9 09-21 00 09 00 12 B8 00 00   ........!.......
1176:0120  46 41 31 2D 31 20 20 20-42 41 4B 20 00 00 00 00   FA1-1   BAK ....
1176:0130  00 00 00 00 00 00 3D 0B-21 00 C0 00 DF 36 00 00   ......=.!....6..
1176:0140  (E5) 41 31 20 20 20 20 20-42 41 4B 20 00 00 00 00   .A1     BAK ....  → 화면 5에서 보는것처럼 화일을 삭제할 경우
1176:0150  00 00 00 00 00 00 2A 0B-21 00 CE 00 12 B8 00 00   ......*.!.......    그 화일의 디렉토리 엔트리의 첫 번째
1176:0160  46 41 31 20 20 20 20 20-48 57 50 20 00 00 00 00   FA1     HWP ....    바이트를 E5H를 써 넣기만한다.
1176:0170  00 00 00 00 00 00 3D 0B-21 00 FD 00 DF 36 00 00   ......=.!....6..    MS-DOS는 이것으로 지워진 화일이라는 것을
1176:0180  46 41 31 2D 31 20 20 20-20 20 20 20 00 00 00 00   FA1-1       ....    인식한다.
1176:0190  (00) 00 00 00 00 00 A7 59-F0 12 0B 01 00 31 00 00   .......Y.....1..  → 첫번째 바이트를 0으로 기록함으로써 여기가
1176:01A0  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   ................    디렉토리 영역의 끝임을 나타낸다.
1176:01B0  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   ................    그러므로 도스 명령의 DIR은 이 첫 번째
1176:01C0  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   ................    바이트가 0이 될 때까지 차례로 디렉토리
1176:01D0  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   ................    영역을 화면에 그 속성에 따라 보여준다는
1176:01E0  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   ................    것을 알 수 있다.
1176:01F0  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   ................
```

적은 갯수의 화일들을 속성 변경시킬 때 유용하다. 이들에 대한 사용법은 아래와 같다.

FA[변경할 화일의 패스와 이름][속성 지정][스위치]

패스를 포함한 화일명에는 와일드 문자인 *과 ?를 사용할 수 있다.

```
속성 지정  /A    화일을 읽고/쓰기가 가능하게
                 지정
           /R    읽기전용 속성을 지정
           /HID  숨기기 속성 지정
           /SYS  시스템 화일속성 지정
                 속성 지정 코드 뒤의 +는 세트,
                 -는 해제를 의미한다.
스위치     /CLEAR 모든 속성을 해제한다.
           /S    서브 디렉토리의 화일도 변경 대상이된다.
           /P    속성변경 화일을 한 페이지 출력한 후
                 대기한다.
           /T    변경된 화일의 갯수만을 보여준다.
```

ATTRIB [+R/−R] [변경할 화일의 패스와 이름]

이 ATTRIB는 읽기전용 속성만 변경이 가능하다.

ATTR [+R/−R][+S/−S][+H/−H][+A/−A][변경할 화일의 패스와 이름]

이 ATTR은 ATTRIB를 모든 속성에 대해 확장한 것이다.

화일의 속성

"낮과 밤은 왜 생겨요?", "비와 천둥

과 번개는 어떻게 생기죠?", "꽃은 어떻게 열매를 맺나요?"라는 호기심어린 질문을 늘 가지고 있는 파브르형 독자께서는 어떻게해서 화일의 속성이 변경되고, 어디에 있는 걸 소트하는 것일까하는 의문을 가질 것이다. 그러면 여기서 이 의문들에 대해서 알아보기로 하자.

MS-DOS에서 사용되는 모든 디스켓에는 디렉토리 영역이라는 것이 있는데, 여기에는 디렉토리명과 화일명, 그리고 그것들의 생성 또는 수정 날짜와 시간이 기록되어 있는 디렉토리 엔트리로 구성되어 있다.

이 디렉토리엔트리는 그림 1과 같이 구성되어 있고, 속성의 변경은 화일명과 화일확장자를 위한 11바이트 다음에 속성 바이트를 조정함으로서, 그리고 화일명의 소트는 이 디렉토리엔트리의 소트로서 가능하다.

이 속성 바이트중 실제 사용하는 것은 하위6개의 비트인데, 각각의 비트에는 MS-DOS가 화일을 다루기 위한 각종 정보가 들어 있다. MS-DOS는 이것으로 화일명과 화일확장자 필드의 것이 볼륨레이블인지, 디렉토리인지 또는 화일인지를 구분하고, 만약 화일인 경우 그것을 화면에 보여주지 않아야하는지 어떤지, 이 화일은 지울 수 있는지 없는지를 알아낸다.

여기서 한가지 주의해야 할 것은 사용자가 직접 디렉토리엔트리의 속성바이트를 수정할 경우 볼륨레이블 지정과 디렉토리 지정을 함께 사용하지 못하는 것 이외에 모든 조합이 가능하다.

그러나 디렉토리 지정과 감추는 화일지정을 함께 세트해서는 안된다. 이는 곧 데이터의 손실이라는 결과를 초래하는 것이 될 수도 있다.

이제는 실제 디스크의 디렉토리 영역을 메모리로 읽어 들인 화면 4를 보면서 좀 더 자세히 접근해보자. 화면 4는 디스크의 디렉토리 영역을 읽은 것으로 소트하지 않은 디렉토리 영역의 예이다.

화면 4에서 디렉토리 영역의 화일명 순과 화면 5의 DIR 명령의 화일명순이 서로 같음을 알 수 있다. 만약 이것을 소트한다면 이 디렉토리 영역도 소트가 됨을 알 수 있다.

앞에서 설명한 디렉토리 소트 유틸리티를 이용해서 직접 소트하고, 화면 4와 같이 디버거를 이용해서 직접 디렉토리 영역을 읽어, 디렉토리 영역들이 소트되어 있음을 확인해 보자. PC

〔화면 5〕 화일의 순서가
디렉토리영역내에서와 같다.

```
B:\>DIR

 Volume in drive B has no label
 Directory of  B:\

DEBUG    BAK    21760  1-01-80   1:12a
DEBUG    GEM    21248  1-01-80   1:15a
2        DBG      409  1-01-80  12:19a
1        DBG     1267  1-01-80  12:16a
HWP      CFG     1054  1-01-80   1:09a
FILEATTR HWP    47122  1-01-80   1:18a
SCR               384  1-01-80   1:04a
FA       HWP    47122  1-01-80   1:19a
FILEATTR BAK    47122  1-01-80   1:15a
FA1-1    BAK    14047  1-01-80   1:25a
FA1      HWP    14047  1-01-80   1:25a
FA1-1           12544  7-16-89  11:13a
       12 File(s)   125952 bytes free

B:\>DEL FA1.BAK
```

프로텍트의 원리

이주헌 / 카멜소프트 특수OA개발부 과장

소프트웨어의 무단 복제를 막으려는 업체와 이를 복사하여 사용하려는 사용자와의 싸움은 오래전부터 지금까지 계속되고 있다. 양자간의 싸움은 한치의 양보도 없이 반전을 거듭하며, 한쪽이 상상을 초월하는 기법을 만들어 상대를 쓰러뜨리면 다른 한쪽은 더욱 기발한 아이디어로 다시 일어선다. 그 방법에 있어서도 소프트웨어적 영역을 벗어나 하드웨어적 방법이 동원되는가 하면 심지어 물리학적, 화학적방법뿐만 아니라 이제는 바이러스를 적용한 생물학적방법까지 적용되는 그야말로 첨단지식이 총동원되는 현대전(?)에 이르게 되었다.

그러나 이 싸움은 방어하는 업체보다는 공격하는 사용자 쪽으로 승리의 깃발이 올라가고 있다. 미국의 경우 소프트웨어업체들이 사용자의 편의를 위해 프로텍트를 푼 소프트웨어를 시장에 내보내고 있다. 이제는 프로그램에 프로텍트를 걸기보다는 프로그램을 구입한 사용자에게 보다 친절하고 정확한 정보와 서비스를 제공할 수 있는 매뉴얼 제작에 힘을 기울여 사용자에게 구매욕을 불러 일으키고 있다.

이러한 공급업체의 사업정책은 사용자들에게도 매우 환영을 받고 있다. 한 예로 요즘 쉐어웨어라는 것이 대단한 인기를 누리고 있는데, 쉐어웨어란 사용자가 프로그램을 복사하여 사용해 본후 자신의 마음에 들면 사용료를 지불하는 것이다.

이제 카피프로텍트는 서서히 사라져 가고 데이터프로텍트 쪽으로 프로텍트 방향이 전환되고 있다. 사회가 점점 정보화 시대로 되어가면서 프로그램의 보호보다는 정보의 근원이 되는 데이터의 보안이 매우 중요하게 되었다. 대부분의 업체들은 엄청난 시간과 노력을 투자하여 얻은 소중한 데이터를 보호하기 위하여 심혈을 기울이고 있다.

독자들은 디스켓을 카피하면서 어떤 디스켓은 카피가 되고 어떤 것은 카피가 되지 않는 것에 대하여 의문을 가졌을 것이다. 이 의문을 조금이나마 해소하기 위해 카피프로텍트에 사용된 기법과 프로텍트 프로그램의 종류 그리고 카피프로그램을 대략적으로 소개하고자 한다.

프로텍트 기법

프로텍트 기법은 여러 종류가 있으나 여기서는 IBM-PC에 제일 많이 적용된 핑거 프린트기법을 소개한다. 핑거 프린트란 말그대로 지문이란 의미이다. 즉 디스크의 특수한 부분에 다른 디스크와 구별되는 흔적이나 정보를 삽입하고 원래의 프로그램에 이 흔적이나 정보가 있는지 검사를 하는 루틴을 추가하여 흔적이나 정보가 있으면 프로그램을 실행하고, 이것이 없으면 프로그램이 실행되지 않는다.

이 방법을 기초로 하여 여러 회사들이 많은 종류의 핑거 프린트를 만들어 냈는데 그것을 대략적으로 분류하면 다음의 4가지로 구분할 수 있다.

첫째 디스켓의 일부분을 기존 포맷과 달리 특수하게 포맷하는 방법이다. 즉 트랙을 더 만들든지 덜 만든다든지 또는 섹터를 더 만들거나 덜 만든다든지 특수한 섹터를 만들어 이상한 ID를 집어 넣어서 디스켓을 포맷하는 기법이다. 프로그램이 실행될때 이 부분을 감지하게 하여 자신이 만들어 놓은 포맷형태와 다르면 프로그램을 실행하지 않는다.

둘째 weak비트를 만든다. weak비트란 일종의 디스크 에러로 디스켓의 일부분을 아주 불안하게 만들어서 매번 읽을 때마다 값이 달라지게 된다. 프로그램에서는 이 부분을 여러번 읽고서 읽은 값을 비교해 값이 서로 다른 경우 프로그램이 실행되고, 같을 경우에는 프로그램이 실행되지 않는다. 일반적인 카피프로그램은 읽을때는 불안하게 읽혀지더라도 쓸때에는 불안하게 읽힌 값을 정확히 기록하는 원리를 이용한 것이다. 과거 대부분의 프로텍트 프로그램은 이 기법을 사용한 것이다.

셋째 두번째의 경우와 비슷한 기법이다. 그러나 이 방법은 보다 강력하게 디스켓에 상처를 낸다. 레이저나 화공약품 또는 망치(?)같은 것으로 디스켓의 일부분에 치명적인 상처를 입힌다. 그리고 상처난 부분의 트랙과 섹터를 기억해두고 그 트랙은 모두 BAD로 만든다. 프로그램이 실행될 때마다 그 트랙을 포맷한 후 상처난 부분을 체크한다. 물리적으로 상처를 입은 부분은 포맷을 하여도 상처난 부분의 BAD는 사라지지 않는 방법을 이용한 것이다.

만일 이 디스크를 카피하여 카피된 디스크가 똑같이 BAD를 가지고 있더라도 이는 물리적인 BAD가 아니기 때문에 프로그램이 실행되면서 그부분을 포맷하면서 그 BAD는 사라져버려 프로그램에서는 BAD를 감지할 수 없다. 이 방법

은 최근에 나온 프로텍트 디스크의 주종을 이루고 있다. 이 방법으로 프로텍트가 걸린 디스크는 카피 프로그램으로는 카피가 불가능하다. 그러나 최근에 나온 여러가지 유틸리티로는 카피를 할 수 있다.

네째 소프트웨어적이 아닌 하드웨어적인 프로텍트 기법이 있다. 이 기법은 특수하게 제작한 하드웨어를 부착했을 때만 프로그램이 실행되도록 프로그램을 구성한 것이다. 즉 프로그램의 일부를 ROM에 집어넣어 그 칩이 꼽혀 있을 경우만 프로그램이 실행되도록 하는 것이다.

프로텍트 프로그램의 종류

외국의 경우 볼트사의 PROLOK 프로그램이 프로텍트의 선봉이다. PROLOK은 디스켓 뒷면에 2개의 작은 상처를 내는 핑거 프린트 기법을 써서 센트럴 포인트 소프트사의 옵션 보드(옵션 보드란 디스크 콘트롤러를 지배하는 보드로서 이를 이용하여 카피를 하면 원본과 똑같이 복사본 디스크를 만들어 준다.그러나 위에서 언급했듯이 물리적인 상처까지 만들어 줄 수는 없다.)를 막아 냈지만 쿼드사의 조그마한 유틸리티에 의해 무너져버렸다. 이 때문에 한때 볼트사와 쿼드사가 법정에 까지 오르게 되었다.

소프트가드사의 SUPERLOK 또한 유명한 프로텍트 프로그램으로 DBASE Ⅲ와 LOTUS, CLIPPER, FRAME WORK 등 유명회사의 프로그램들을 프로텍트를 걸었으나 이것 역시 쿼드사에 의해 무너졌다.

또 Az-TECH SOFT사의 EVERLOCK이 있다. 이 프로그램은 소프트웨어적 프로텍트를 쓴 것 중에는 상당히 우수한 것이나 국내에서는 해커들에 의해 산산 조각이 났다. 최근들어 버전2.0을 내놓았지만 고전을 면치못하는 것 같다.

국내에서 프로텍트를 가장 먼저 상업화시킨 것은 가인시스템의 하드록 디스켓일 것이다. 이는 위에서 언급한 핑거프린트의 세번째 기법에 속한다. 하드록은 카피 프로그램으로는 카피가 불가능하지만 프로그램을 뚫고 들어가면 의외로 프로텍트가 쉽게 풀린다는 소리가 있다. 아마 같은 회사로 기억이 되는데 하이록이라는 디스켓도 하드록과 큰 차이가 없었다.

또 벤엘컴퓨터의 B-LOCK 디스켓도 위의 세번째 기법과 비슷한 기법이 적용되었다. 특이한 점은 자체적으로 해커를 보안하는 루틴이 있어 만일 해커가 프로그램 내부에 손을 대면 프로그램이 시스템에 영향을 줘서 심할•경우는 시스템이 파손되는 경우도 있다. 역시 카피 프로그램으로는 카피가 불가능하다. 그밖에 프로텍트 프로그램이 있는데 이것은 카멜소프트에서 제작된 PANDORA-F, H, D, S이다. 이 프로그램에서 좀 특이할만한 것은 PANDORA-H와 D이다. PANDORA-H는 하드디스크 전용 프로텍트이고, PANDORA-D는 데이터 전용 프로텍트이다. 또 PANDORA-S라고 해서 국내에선 처음으로 사용자가 개발한 프로그램에 맞춰서 프로텍트를 걸어주는 프로그램도 있다.

대표적인 카피 프로그램

카피 프로그램이라 하면 제일 먼저 생각나는 것이 센트럴 포인트 소프트웨어사의 COPY Ⅱ PC이다. 이 회사는 PC-TOOLS 로 잘 알려져 있다. 특히 얼마전에 나온 COPY Ⅱ PC 5.0중 NOGUARD.COM이라는 프로그램은 여러가지 프로텍트을 제거하는 프로그램으로서 상당히 우수하다.

이것은 PROLOK과 SUPERLOK 또 EVERLOCK 등 여러가지 프로텍트를 제거해준다. 이 프로그램중에 NOKEY. COM이라는 프로그램은 하드 디스크 사용시 키디스켓을 넣지않고도 가능하도록 해주는 프로그램이다.

그러나 이 보다 더 강력한 카피 프로그램은 쿼드 소프트웨어의 COPYWRITE로서 소프트웨어적으로 걸린 모든 프로텍트를 제거한다고 해도 관언이 아니다. 필자가 한때 프로텍트 프로그램을 개발할때 이 COPYWRITE때문에 고생한 것을 생각하면 치가 떨릴 정도이다. 필자가 본 소프트웨어적 프로텍트 중에 EVERLOCK 2.0을 제외하고는 이 프로그램에 의해 모두 카피가 되었다.

가장 최근의 프로텍트 프로그램을 제외하고는 거의 다 카피가 된다. 특히 게임을 카피할 때의 성능은 만점이다. 그러나 이 프로그램 역시 위 프로텍트 기법중 3번째를 극복할 수는 없다. 이에 대해서는 쿼드사가 여러가지 유틸리티를 제공하고 있다. 그외 여러가지 카피 프로그램이 있으나 대동소이하다.

데이터 프로텍트

개인용 컴퓨터는 대형 컴퓨터와는 달리 데이터를 복사하기 쉽다. 대부분의 화일들은 하드디스크내에 보관하는데 누구나 중요한 화일들을 주인이 없을때 와서 아주 손쉽게 가져갈 수 있다. 또 네트워크나 모뎀을 이용한 통신상에서도 도난을 당할 우려가 있다. 아직까지 우리나라에서는 데이터 프로텍트에 대해서는 커다란 관심을 보이지는 않지만 외국의 경우에는 이것이 상당한 관심사로 떠오르고 있다.

데이터에 프로텍트를 거는 방법중에 가장 쉬운 방법이 데이터를 암호화하는 것이다. 이것은 아주 먼 옛날부터 사용해온 방법이다. 이 방법은 남에게 데이터가 노출되어도 암호를 모르면 그 데이터는 소용이 없다.

현재 거의 대부분의 데이터는 이 방식으로 보호가 되어 있지만 데이터를 사용할때 마다 암호화시키고 풀고해야 하므로 일이 매우 번거롭다. 또 이 과정에 소요되는 시간도 무시할 수 없다. 그래서 하드웨어적인 암호화보드가 나왔는데 이 보드는 데이터를 디스크에 기록할 때 자동적으로 암호화를 시켜주고 읽을때 자동적으로 해독을 해준다. 또 이 과

정이 하드웨어적으로 처리되어 속도가 빠르고, 사용자는 이 과정을 거의 느끼지 못한다. 암호화하는 방법은 가장 쉬우면서도 강력한 방법이다.

다음으로 많이 사용하는 방법이 패스워드를 이용하는 것이다. 시스템이나 데이터가 들어있는 디렉토리를 패스워드로 묶어 패스워드를 알고 있는 사람만이 그 데이터를 사용할 수 있게 만든다. 그러나 개인용 컴퓨터에서는 적절한 방법이 못된다. 이 방법은 다른 사람이 패스워드를 제어하는 프로그램을 풀 가능성이 있기 때문이다. 이것을 방지하려면 패스워드와 암호화를 병행하는 것이 좋다.

또 다른 방법으로는 하드디스크에 있는 데이터를 교묘한 방법으로 감추는 것이다. 데이터를 어떠한 특수한 디렉토리에 담아두고 그 디렉토리를 숨기는 방법이다. 이외에도 최근에는 아주 특수한 방법들이 많이 개발되고 있는데 가장 두드러진 일예가 바이러스이다. 앞으로는 데이터 프로텍트 이상으로 바이러스 감염에도 주의를 기울여야 한다.

맺으며

카피프로텍트는 해커에게 있어서는 언제나 즐거운 연구대상이 되곤 하였다. 필자 또한 한때는 프로텍트를 걸로 풀고 하는 일이 매우 흥미있는 관심의 대상이 되었다. 허나 지금 와서 돌아보면 그 많은 시간과 노력을 다른 곳으로 기울였다면 어떠 했을까 하고 생각해 본다.

요즘 나오는 프로텍트들은 플로피와 하드디스크 양쪽다 적용될 수 있도록 개발되어 있다. 프로텍트가 걸린 디스켓을 구입하면 인스톨이란 과정으로 프로그램을 하드디스크에 넣을 수 있는데 인스톨을 실행시키면 플로피 디스크의 프로그램은 지워지거나 더 이상 인스톨이 되지 않는다.

하지만 요즘처럼 바이러스가 들끓고 있는 세상에서 하드디스크가 언제까지나 안전하리라는 보장이 없다. 또 사용자의 부주위나 기계적인 결함에 의하여 프로그램이 깨질 우려가 있다. 프로그램을 판매한 업체는 사용자들에게 언제나 주의하라는 이야기만 할뿐 사후 대책에 대해서는 언급이 없다.

이글을 끝까지 읽으주시느라 수고하신 독자를 위해 조그마한 프로그램을 만들어 봤다. 이 리스트 1은 인스톨의 횟수를 자유롭게 해주는 것이다. 원리는 아주 간단하다. 프로텍트의 인스톨 프로그램은 플로피의 프로그램을 모두 하드로 옮긴후 자기볼일(?)을 본다. 자기 볼일이란 프로텍트를 하드로 옮기거나 플로피디스크에 더 이상 인스톨을 못하도록 조작을 가하는 일이다. 플로피에 조작을 가한다는 의미는 플로피에 어떠한 정보를 WRITE한다는 의미가 된다. 그렇다면 답은 나왔지 않은가?

즉 플로피에 WRITE를 못하게 하면 된다. 그런데 플로피에 WRITE PROTECT붙이는 방법은 컴퓨터가 디스크 에러를 발생시키므로 컴퓨터를 속이는 것이 필요하다. 롬 바이오스 중 INT 13이 디스크 입출력을 담당하는 부분이므로 이부분을 가로채서 실제로는 디스크에 WRITE동작을 하지 않았지만 WRITE를 한 것으로 속이는 램 상주 프로그램을

〔리스트 1〕 인스톨을 자유롭게 해주는 프로그램

```
CODE SEGMENT
     ASSUME CS:CODE
     ORG 100h

START:
       JMP INIT                        ; 초기화 루틴으로 간다.

OLD_INT13   DD  ?

; 새로운 INT 13 루틴

NEW_INT13:
       PUSH AX                         ; 레지스터 값을 보관한다.
       PUSH BX

       CMP  DL,0                       ; 드라이브 A 를 사용하는가 ?
       JNE  CONTINUE                   ; 아니면 CONTINUE 로 감.
       CMP  AH,03                      ; 드라이브에 WRITE 하는가 ?
       JE   QUIT                       ; 맞으면 QUIT 로 감.
       CMP  AH,05                      ; 드라이브에 FORMAT 하는가 ?
       JE   QUIT                       ; 맞으면 QUIT 로 감.
CONTINUE:
       POP  DX                         ; 레지스터를 복귀한다.
       POP  AX
       JMP  OLD_INT13                  ; 원래 INT 13 을 수행한다.

QUIT:
       POP  DX                         ; 레지스터를 복귀하고
       POP  AX                         ; INT 13 을 수행하지 않고
       IRET                            ; 끝낸다.
INIT :
       MOV  AX,3513h                   ; INT 13 의 벡터 값을 읽는다.
       INT  21h
       MOV  WORD PTR [OLD_INT13],BX    ; 원래의 INT 13 의 벡터 값을
       MOV  WORD PTR [OLD_INT13+2],ES  ; OLD_INT13에 보관한다.

       MOV  DX,OFFSET NEW_INT13        ; INT 13 의 벡터값을
       MOV  AX,2513H                   ; NEW_INT13 으로 바꾼다.
       INT  21H

       MOV  DX,OFFSET INIT             ; 램 상주 시킨다.
       INT  27H

CODE   ENDS
       END  START
```

작성하면 된다.

프로그램을 정확히 입력했으면 어셈블러로 컴파일하고 링크시킨후 EXE2BIN.EXE를 이용하여 COM화일로 만든다. 사용법은 아주 간단하다. 이 프로그램을 실행시킨후 프로텍트된 디스크에 담긴 설명서 그대로 인스톨시키면 된다. 단 기억해야 할 점은 프로텍트된 디스크는 인스톨이 가능한 디스켓이어야 한다는 것이다. 그리고 만일을 대비해서 프로텍트가 걸린 플로피 디스크에 WRITE PROTECT를 붙이고 사용하는 것이 좋다.

예를 들어 EVERLOCK이 걸린 프로그램일 경우에는 이 프로그램을 실행시킨다. 이 프로그램이 실행된 후에는 A 드라이브에는 아무 것도 WRITE할 수 없다. 그후 프로텍트가 걸린 디스켓을 하드디스크로 인스톨시키면 된다. 인스톨이 끝나면 리셋을 눌러서 드라이브 A가 WRITE 가능하게 한다. 프로텍트 디스켓은 나중에 프로그램이 깨졌을 경우를 생각해서 안전한 곳에 보관한다.

이 프로그램이 제발 다른 용도로 사용되지 않기를 바란다. 이 프로그램은 단지 인스톨만 자유롭게 해줄뿐이지 다른 프로그램에 전혀 손을 대지 않는다. 따라서 사용자가 프로그램을 다른 곳으로 옮길 경우 시리얼 넘버라든가 기타의 정보들이 그대로 옮겨지므로 이에 따른 책임은 사용자가 지게 된다. 칼자루는 쥐고 있을때 제일 주의하여야 한다. PC

국내소식

—과학기술처

국책연구개발사업 대형과제중심으로 전환

과학기술처는 내년부터 국책연구개발사업의 수행방향을 대형과제 중심으로 전환하기로 했다. 그리하여 10대 중점사업분야를 선정하는 동시에 국책연구개발사업을 종합적으로 기획, 관리할 수 있는 국책연구개발 사업단을 설치, 운영키로 했다.

또 첨단기술 분야의 연구개발활동을 적극 활성화하기 위해 정부지원 연구개발비 규모도 1천6백억원으로 확보하기 위해 관계당국과 협의할 방침이다.

과기처가 중점투자키로 한 분야는 정보산업, 메카트로닉스, 신소재, 생명공학, 항공, 우주, 해양기술 등이며 이를 위해 관계부처, 학계, 산업계, 연구계 인사가 참여하는 국책연구 개발사업단이 설치되어 관련업무를 기획하고 관리하도록 할 계획이다.

—과학재단

각 대학별 우수연구센터 선정 공모

과학재단(이사장 : 정근모)은 각 대학별로 기초 연구센터를 선정하여 각 대학의 자연계 학과가 공동으로 이용할 수 있도록 하였다. 이에 따라 과학재단은 다양한 기초연구 수요를 효과적으로 관리하고 잠재적인 지역적특성 등을 적극적으로 활용할 수 있도록 하기 위해 각 대학의 연구센터 설치 및 운영을 지원하기로 했다.

따라서 새로운 이론정립은 물론 심층적인 연구 등 기초과학을 주로 연구하는 과학연구센터(SRC)와 산업관련 기초 기술연구를 주로 하는 공학연구센터(ERC)가 설치될 예정이다.

이 연구센터의 선정은 전국대학을 대상으로 공모하여 과학재단의 심사로 선정되며, 신청은 7월까지 지원은 11월부터 할 계획이라고 한다.

—체신부

4차 VAN 개방조치발표

체신부는 정보통신회선 사용 제한을 대폭 완화하는 내용의 4차 VAN 개방조치를 발표하고 이를 7월 1일부터 적용하였다. 현재 정보통신회선 사용시 단순 정보검색 및 정보처리서비스만 제공해온 정보통신 역무제공업자의 사업범위가 대폭 확대되어 여러 계층을 대상으로 한 정보교환 및 통신매개행위 등을 할 수 있게 되었으며, 그룹계열사나 거래고중 이용율이 10분의 1이상인 자로 이용범위를 확대함으로써 그룹 VAN도 활성화 될 것으로 전망된다. 또한 민간 VAN사업자의 회선이용을 9600 bps에서 다중화장치로 64Kbps이상까지 확대, 회선의 이용효율을 극대화하였다.

이로써 민간 기업의 부가통신망사업 참여가 본격적으로 허용되어 역무제공업자의 정보교환 등을 활성화 시키는 계기가 될 것으로 내다보고 있다.

그러나 이번 개방조치는 민간 VAN 허용이라는 내용을 포함하였지만 모두 국내회선 부분에만 국한시켰다는 지적이 대두되었다.

현재 국내 부가통신사업자는 12개 그룹 VAN업체와 125개의 정보통신 역무제공업체가 있다.

—서울시 경찰국

경찰업무 전산화 추진

서울시 경찰국은 경찰사무행정의 과학화 추진의 일환으로 왕컴퓨터 코리아의 VS시스템과 통합화상 정보관리시스템을 도입하여 경찰내부업무의 전산화를 추진하고 있다.

서울경찰이 보유하고 있는 2만여대의 무선장비관리의 자동화 및 경찰급여관리 전산화 등 행정업무의 자동화에서 치안요도, 범죄자 통합관리, 지휘보고, 112신고 자동즉응시스템 등 범죄수사관리업무까지 자동화할 이 경찰업무 전산화는 오는 92년에 완료될 예정인데 현재 첫단계로 범죄수사에 필요한 범죄수법 및 피의자 등 4만8천매에 달하는 각종 인적, 물적자료를 화상자료로 전산화한 것을 비롯하여 112신고 자동즉응 시스템 구축작업을 진행중에 있다.

—한국정보과학회

제6회 정보분야 논문발표회

한국정보과학회는 지난 6월23일 연세대학교에서 제6회 정보과학분야 박사학위 논문발표회를 개최했다.

이번 발표회에서는 정보산업에 종사하는 각계의 인사가 모인 가운데 서강대 전자계산학과의 오경환교수의 범용



Fuzzy 전문가시스템 설계에서의 새로운 방법론과 부산대 전자계산기공학과 이정태 교수의 음성의 묵음시간을 활용한 데이터통신 프로토콜의 설계에 관한 논문 등 20개의 논문이 발표되었다.

동학회는 회원들의 학술연구 활동을 폭넓게 수용하여 회원간의 정보교환과 유대를 도모하고자 정보과학 분야의 박사학위 논문 발표회를 정기적으로 개최하여 왔다.

—전산망조정위원회
교육용컴퓨터 표준규격 확정

전산망조정위원회는 교육용 컴퓨터의 16비트 확정과 함께 본격화되기 시작한 컴퓨터교육의 실습기종을 정부표준기기와 호환성을 가지는 규격으로 확정하였다.

이번에 확정된 교육용 컴퓨터의 표준규격은 교사용이 80286CPU를 채택한 AT호환기종, 학생용이 8088CPU의 XT 호환기종으로 결정되었다.

교육용 컴퓨터의 표준규격은 국가표준기기의 기본규격을 주축으로 컴퓨터 교육에 꼭 필요한 조건을 첨가하여 마련된 것으로, 의무적으로 시력보안용 필터를 부착하고, 한글이 16×16 도트로 표현되도록 허큘리스보드를 기준으로 하며 VGA보드의 채용도 가능하도록 하였다. 모니터는 교사와 학생용 모두 모노크롬으로 규정하였으며, 키보드는 정부표준의 한글 배열로 설계된 것으로 한글표시가 영문보다 크게 인쇄된 제품으로 규정하였다.

학생용은 주기억 용량이 512KB 이상으로 5.25인치 플로피디스크 1대 이상을 갖춘 시스템으로, 교사용은 주기억 용량이 1MB 이상으로 20MB급 이상의 하드디스크와 3.5인치, 5.25인치 플로피 디스크를 각각 1대씩 장착한 컴퓨터로 결정했다.

그리고 교사용 시스템에는 132컬럼 프린터와 공중통신망에 연결할 수 있도록 하는 다이얼업 모뎀을 각각 부착하여 보급하도록 하였다.

—체신부
고유업무 분담제 추진

체신부는 컴퓨터통신의 활성화를 위해 전기통신공사, 데이타통신, 전자통신연구소, 통신진흥, 정보문화센터 등 5개기관별로 역할을 분담하여 추진키로 했다.

체신부가 컴퓨터통신 추진계획을 마련함에 따라 우선 통신공사는 단계별 PC통신 활성화 계획과 시행, 통신방식 표준화, 통신서비스 관련 주요자료의 데이터베이스, 핵심기술 연구개발 사업 등을 전담할 계획이다.

데이타통신은 데이터베이스와 관련기술개발 및 표준화 데이터베이스 구축 대상분야와 우선순위 선정 등을 주로 담당할 방침이며, 전자통신연구소는 단말기 표준사양 작성 및 연구개발, PC 통신용 어댑터와 소프트웨어개발, 각국의 단말기 공급전략에 관한 검토작업 등을 맡기로 했다. 또한 정보문화센터는 정보이용 활성화를 위한 각종 홍보 활동 등을 추진키로 했다.

체신부는 기존 전화망을 이용하여 개인용 컴퓨터간의 통신과 전화가능한 컴퓨터통신의 실용화를 앞당기기 위해 7월부터 퍼스널컴퓨터의 통신용 부가장치 개발에 이어서 내년 하반기부터 상용서비스를 개시할 방침이다.

—한국 전자통신연구소
4MD램 개발성공

정부와 민간 공동연구로 추진된 4MD램 개발사업이 초기의 목표를 상회하는 실적을 거두고 종결되었다.

한국전자통신연구소는 85년 9월부터 89년 4월까지 3년간 수행된 반도체 공동연구를 통해 산업계는 선폭 0.8미크론의 4MD램 생산기술을 확립하고, 최근까지 20.39%의 수율을 달성하였으며, 115건의 특허출원실적을 이룩하였다고 밝혔다.

총 연구책임자인 한국전자통신연구소 반도체기술 연구단장 김정덕씨는 이번 4MD램의 개발성공으로 반도체 분야의 자체개발 능력을 확보함과 동시에 국제경쟁에서도 선두대열에 동참할 수 있게 되었으며 반도체 공동연구사업에 의해 4MD램급 이상의 설계능력을 수립하였다고 한다.

이번 4MD램의 개발성과는 0.8미크론 이하급의 미세선폭가공기술 등을 축적할 수 있고 각 기업과 연구소·학계를 통틀어 457건의 부수적인 특허가 출원되었으며 제품설계기술과 관련한 문서의 정립과 산업·연구계를 통틀어

총 2천여명에 달하는 전문연구인력의 확보 효과를 얻게 되는 계기가 되었다.

—한국 IBM
소프트웨어 엔지니어링 일차 마무리

한국 IBM은 한국과학기술원의 시스템공학센터와 함께 7월 6일 인터콘티넨탈호텔에서 두기관의 제1차 소프트웨어 엔지니어링 공동 협력사업의 성공적인 수행을 기념하는 축하연을 가졌다.

이날 행사에는 과학기술처장관, 체신부장관, 과학기술원장 등 각계인사 1백여명이 참석하였으며, 협력사업 경과보고, 새로운 사업에 대한 합의서 교환, 축하연 등의 순으로 진행되었다.

두 기관의 공동협력사업은 지난 85년 7월 12일 발족한 소프트웨어 엔지니어링부(SEC)로 국내 소프트웨어 생산성 재고 및 향상된 소프트웨어 개발환경의 조성 및 지원의 목적하에 협력기한 4년으로 추진되어 그동안 소프트웨어 엔지니어링 교육과정을 통해 4천여명의 수료생을 배출하여 국내의 부족한 소프트웨어 전문인력 양성에 도움을 주었고, 이 사업과 관련하여 시스템공학센터는 지난 85년부터 국책 연구과제로 프로토타입 툴 연구, S/W 조직, 역공학, 소규모 SDE(Software Development Environment)에 관한 연구를 수행해 왔다.

그동안 4년간에 걸친 1차 협력사업 종료후 시스템공학센터와 한국 IBM은 소프트웨어 엔지니어링 교육과 관련하여 향후 2년간 전문강사들에 대한 교육에 힘쓸 예정이라고 한다.

—한국전기통신공사
교환기 공급회선 278만1천 회선으로 확대

한국전기통신공사는 내년도 교환기 공급물량을 올해보다 49%가 늘어난 278만1천 회선으로 최종 확정하였다. 한국전기통신공사의 국설 교환기 공급계획에 따르면 이는 당초 계획했던 160만 회선보다도 늘어난 규모로 국설 교환기 공급이래 최대치를 갱신하는 수치이다.

이렇게 예상보다 공급물량이 크게 늘어난 것은 도시영세민과 농어촌지역 주민들의 전화수요가 급증하는데 따른 전화기 적체현상 해결방안의 하나로 지적되고 있다.

국설교환기 공급내역을 업체별로 보면 금성반도체가 34.6%로 으뜸이며, 삼성전자가 34.6%, 동양전자통신이 18.8%를 차지하는 것으로 집계되었다.

한편 올해 43만대인 전전자교환기 TDX도 내년 133만 회선으로 늘어남에 따라 전체 공급량의 절반을 차지하게 된다. TDX 공급량이 전체의 86.6%를 차지함으로써 통신망의 디지탈화율도 크게 진전될 전망이다.

—대우전자
교육용 PC 아이큐-슈퍼 시판

대우전자는 확충되는 교육용 컴퓨터 시장과 고기능을 요구하는 비지니스분야에서 시장점유율을 높여 나가기 위한 일환으로 최근 XT기종 3가지와 AT기종 4가지 모델을 개발하여 PC사업을 강화하고 있다.

지난 7월부터 본격적으로 판매된 이들 컴퓨터는 '아이큐-슈퍼'로 명명된 교육용 컴퓨터 CPC-4000LC를 비롯, 행정전산망용 CPC-4000S, CPC-4020S와 AT 컴퓨터 CPC-6220S외 3가지 모델 등으로, 이 모델들이 판매되면 국내 PC시장이 더욱 치열해 질 전망이다.

특히 교육용 컴퓨터 '아이큐-슈퍼' CPC-4000LC는 주기억용량 512KB에 플로피디스크 2대가 내장되어 있고 MDA, CGA 모드를 지원한다.

한편 대우전자는 중학교 교과중 영어, 수학, 과학 교육용 등의 CAI 프로그램을 개발하여 보급할 예정에 있으며, MS-DOS, 한글MS-DOS, 워드프로세서 나랏말씀II, 메뉴 제너레이터, DACOM-NET EMULATOR, GW-BASIC 소프트웨어 등을 기본으로 공급할 방침에 있다.

—한국 전자통신연구소
종합서비스체계 ENS 추진 제시

한국전자통신연구소는 대외개방, 경쟁체제 도입 등 통신사업 분야에서 경쟁력 확보를 위한 ENS(Enchanced Network Service)추진 방안을 한국전기통신공사에 제시했다.

한국전자통신연구소가 제시한 ENS는 기존통신망(PSTN)에 통신, 정보처리 기능을 추가하여 음성, 비음성 서비스를 제공하는 미래지향적 종합서비스 제공체계로 통신공사가 ENS를 추진하면 단기적으로 기본통신망의 운영제도와 처리기능을 부가할 수 있으며, 장기적으로 ISDN으로 발전하는 통신망과 새로운 서비스를 통합하여 경영효율을 극대화할 수 있다.

—삼성데이타시스템(주)
GL:M 국내 독점공급 개시

삼성데이터시스템(주)는 미국의 "맥코맥&도지"사와 재무관리·회계정보시스템인 GL:M(General Ledger:Millennium)에 대한 국내 독점공급계약을 체결하였다.

종합적인 재무·회계시스템인 GL:M은 고정 및 변동예산 편성이 가능하고 연결재무제표의 작성, 다양한 원가배분 기능을 가지고 있으며 재무계획과 재무모델을 그래프로 나타낼 수 있다.

한편 삼성은 효율적인 응용시스템 개발 툴인 밀레니움 SDT 등 각종 응용 S/W 패키지에 대한 계약도 체결하였다.

밀레니움 SDT는 밀레니움 환경하에서 기존 시스템의 수정 및 새로운

시스템의 작성을 보다 빠르고 쉽게 수행할 수 있다.

밀레니움은 4세대 언어개념을 적용하여 어플리케이션의 유지·보수가 용이하고, 서로 다른 데이터간이나 외부화일과의 효율적인 인터페이스가 가능한 우수한 특징을 가지고 있으며, IBM 370, IBM 43××시리즈, IBM 30××시리즈 및 호환기종에서도 사용할 수 있다.

—대명컴퓨터
레이저 포토플로터 사업강화

대명시스템(대표:오철성)은 최근 미옵트로닉스사가 개발한 레이저 포토플로터의 국내 공급을 늘려나가는 한편 A4 크기 필름을 2분 이내에 플로팅해 낼 수 있는 고속의 신제품을 출하할 계획을 세우는 등 포토플로터 사업을 강화하기로 하였다.

이러한 계획의 일환으로 L2620 및 L2420 레이저 플로터를 공급할 예정인데 이는 초당 44라인의 플로팅 속도로 최대 20인치×26인치 크기의 필름을 제작할 수 있는 것으로 5MIPS·32비트 컴퓨터에서 사용할 수 있다. 이와 함께 스위스 글래서사의 플래서베드타입인 DP3545와 LED어레이를 이용한 래스터 방식의 갤럭시 400 신제품도 함께 늘려갈 방침이다.

—대우전자
서울의사회에 컴퓨터와 병원용 소프트웨어 공급

대우전자는 서울의사협회에 병원용 코로나 컴퓨터(PC-4000 II)와 병, 의원용 소프트웨어인 DMICS(Drain Medical Information Control System)를 공급하기로 계약하였다.

이 제품의 특징은 조작과정을 단순화하고 외래, 입원, 의료보호의 보험처리를 비롯해 자동차 보험관리, 산업재해 보험관리, 지역별 환자관리, 차트관리, 일일수입 및 수납현황, 약품재고관리, 수술환자관리, 통계자료 등 병원업무 전산화를 꾀한 것이다.

—(주)삼보컴퓨터
트라이젬 데이타 시스템사 설립

(주)삼보컴퓨터(대표 : 김종길)는 일본의 21개 소프트웨어회사 단체인 21세기 클럽과 양국간의 소프트웨어 개발 및 판매를 위한 합작회사로 트라이젬 데이타 시스템사(TriGem Data System Inc.)를 국내에 설립키로 하였다.

이에 따라 지난 7월 12일 일본에서 삼보측의 이용태회장과 이덕순 소프트웨어 담당사장, 21세기측의 (주)Japan Computer Engineering사의 石田辛男사장외 21명의 일본측 대표가 참석한 가운데 합작회사 설립 조인식을 가졌다.

—삼성전자
벨지움의 BTM사와 기술협력협정 체결

삼성전자는 정보통신업체인 벨지움의 BTM사와 통신분야, 정보시스템사업 및 특수반도체 등의 사업분야에 대한 기술협력 공동연구개발 및 세계시장의 공동개척을 위해 기술협력 협정을 체결했다.

이로써 양사는 ISDN용 교환기, 통신기기, 송신설비 등 통신분야와 정보기기 사업 전반에 걸쳐 하드웨어, 소프트웨어 기술, 시스템 및 그와 관련된 각종 첨단기기 및 부품 뿐만 아니라 반도체소재를 포함하여 앞으로 첨단부품들에 대한 기술이전, 공동개발, 제품생산 및 판매협력을 통해 세계 정보통신 시장을 공동으로 개척하게 된다.



이를 위해 양사는 기술정보 및 제품 샘플의 교환, 전문기술 인력의 상호교류, 상호 관심 제품에 대한 공동개발 및 합작판매회사 설립을 위해 양사의 임원급을 위원으로 하는 협력위원회를 구성하여 세부적인 사업계획을 수립하기로 했다.

—삼성종합기술원
데스크탑 이미지 스캐너 개발

삼성종합기술원(대표 : 소진화)은 컴퓨터를 연결하여 사진, 도면, 문서 등을 컴퓨터에 입력, 저장할 수 있으며 프린터로 입력된 정보를 인쇄할 수 있을 뿐만 아니라 PC-FAX의 입력장치로서 화상전송도 가능한 이미지 스캐너를 개발하였다.

이번에 개발된 이미지 스캐너는 스캐닝방식이 원고고정형(FLAT-BED TYPE)으로 낱장의 문서뿐 아니라 제본된 문서까지도 자유롭게 입력할 수 있게 하였으며, 1인치당 300도트와 16단계의 명암을 구분할 수 있는 고해상도와 최소 스캐닝 시간 14초를 실현했다.

입력 가능한 원고의 크기는 최대 A4에서 명함크기까지 가능하며 이용자가 원하는 일부분의 화상만을 입력할 수 있는 기능도 갖추고 있다. 또한 컴퓨터 기억장치의 효율적인 활용과 처리시간을 단축하고 팩시밀리와 통신기능을 위하여 화상데이터를 압축처리할 수 있는 기능 및 널리 보급된 각종 퍼블리싱 소프트웨어, 기타 OCR 소프트웨어 등과 데이터 호환성을 유지하기 위한 다양한 출력화일을 제공하고 있다고 한다.

이미지 스캐너의 이용분야는 현재 반이상이 전자출판분야이나 향후에는 다목적 응용분야와 PC-FAX, CAD분야가 70% 이상을 차지함으로써 시장이

급격히 신장될 것으로 전망된다.

삼성종합기술원은 이번 개발을 통하여 빛신호를 전기신호로 바꾸는 광전환기술, 이미지속의 글씨를 인식하는 문자인식기술, 디지탈 이미지 프로세싱기술, 방대한 양의 데이터를 압축하는 데이터압축기술, 복원기술 등을 축적하였으며, 이러한 기술을 삼성전자에 이전시켜 제품을 생산할 계획이다.

—쌍용컴퓨터
그래픽 가능한 WP 개발

쌍용컴퓨터는 문서편집은 물론 원, 사선, 직사각형 등 32가지의 모양을 자유자재로 그릴 수 있도록 그래픽지원이 추가된 화랑워드프로세서를 개발했다.

이번에 개발된 워드프로세서는 드래곤 386 등 각종 386퍼스컴과 EGA 그래픽보드를 내장하고 있는 IBM XT/AT 호환기종에서 사용가능하며, 화면상에 편집된 내용과 출력 내용이 일치하도록 설계되어 있어 사용자가 손쉽게 사용할 수 있다.

또한 한글과 한자, 일본어를 지원하고 있으며, 문자체는 고딕체, 명조체, 샘물체 등 3종류의 표현이 가능하고, 그래픽으로 32가지 모양의 데이터를 자유롭게 입출력할 수 있다.

한글은 2바이트 조합형과 완성형이 지원되며, 폰트디자인 툴을 사용하여 문자를 확대, 축소할 수 있어 도표를 많이 사용하는 보고서나 간단한 책자의 편집에 적합하다.

—에이직코리아
AT용 니트칩세트 개발성공

하드디스크 전문 중소기업체인 에이직코리아는 퍼스널 컴퓨터의 핵심기술중 하나인 AT용 니트칩세트 개발에 참여하여 국산화에 성공하였다. 이 AT용 니트칩세트는 지난 87년 미국의 시스로직 테크놀러지사와 공동 개발을 시작한 것으로서, 지난달 시험을 마치고 6월부터 대량 생산체제로 돌입, 올해에는 미국지역에 4만세트, 아시아지역에 2만세트를 공급할 계획이라고 한다.

국내에는 처음으로 20MHz용으로 설계되어 개발된 니트칩세트는 88C211 CPU 및 버스콘트롤러, 88C212 메모리 콘트롤러, 88C215 데이터, 어드레스 버퍼 등 3개의 칩과 1개의 범용칩으로 구성되어 있다.

—퍼스컴 제조업체
마더보드 공급활기

최근 퍼스컴 XT, AT, 386의 마더보드 시장이 수요가 급증하게 되자 영세 퍼스컴제조업계들에 의해 형성되었던 마더보드 시장이 전문공급업체를 중심으로 활기를 띠고 있다.

공급업체들은 특히 저가형 퍼스컴을 조립하는 50~60개의 영세 업체들에게 한꺼번에 10~100개의 제품을 OEM방식으로 제작해 주거나 일괄계약에 의해서 관련부품까지 공급하는 방식을 취하고 있다.

마이크로텍은 8~12MHz의 마더보드를, 글방컴퓨터는 25MHz 및 33MHz의 386보드를 공급하고 있으며, 립텍컴퓨터는 10MHz의 XT보드, 12, 16, 26MHz의 AT보드를, 석정전자는 15MHz의 XT보드, 프린스컴퓨터는 4.77~8MHz의 XT보드를 저가로 공급하고 있다. 업계는 올연말을 기점으로 AT 및 386 퍼스컴 보급이 크게 늘 것으로 보고 국내시장 규모가 월 1천5백개 수준이 될 것으로 전망하고 있다.

—한국데이타통신
9월부터 저가형 단말기 보급 예정

지난 6월 24일 한국데이타통신은 자사의 부가통신서비스 이용촉진과 보급을 위해 저가형 단말기규격안을 마련하여 관련업체를 대상으로 제안설명회를 가졌다. 이 제안설명회 후 현대, 금성, 삼보 등 8개 업체가 제안서를 제출하여 이 업체들을 대상으로 선정작업에 착수하였는데, 한국데이타통신은 6월중에 1차로 2~3개 업체를 선정하고 다시 시제품검사를 통해서 8월중에 최종업체와 제품을 선정한 뒤 9월부터 본격적으로 보급에 들어갈 계획이다.

이 검색전용단말기는 1200bps 비동기 다이얼모뎀을 내장하여 천리안II정보를 이용할 수 있을 뿐만 아니라 프린터와도 연결이 가능할 수 있도록 되어 있다.

데이타통신은 DNS가입자를 대상으로 보급할 이 단말기의 공급가격을 20만원대로 예상하고, 연내에 3천대, 내년상반기까지 7천대 가량을 보급할 예정에 있다.

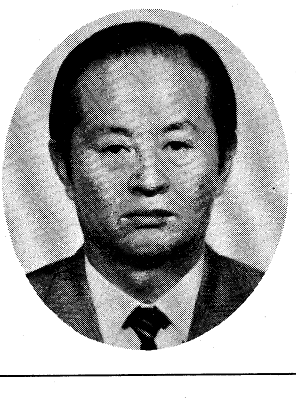
이 공중검색용 저가형 단말기가 보급되면 단말기 구입비용을 줄일 수 있어 공중 데이터베이스 검색등 컴퓨터통신 이용이 현재보다 더 활발해질 것으로 전망된다.

—왕컴퓨터코리아
정우진이사 전무로 승진 발령

왕컴퓨터코리아는 지난 7월 1일자로 기술부 정우진 기술이사를 영업, 기술 및 마케팅 담당 전무이사로 승진 발령했다.

정우진 전무는 한국IBM의 기획관리 실장을 거쳐 지난 87년 11월부터 왕컴퓨터코리아의 기술이사로 재직했다.

"이미지 프로세싱 분야에서 통합화



상 정보처리시스템(WIIS) 판매에 주력하여 화상, 데이터, 문서, 음성 등 모든 형태의 정보를 통합처리할 수 있는 시스템 판매자로서의 왕의 이미지를 확고히 하겠다"고 영업전략을 밝힌 정우진 전무는 영업 뿐만 아니라 고객지원 활동도 다양화하여 고객만족도 증진을 위하여 노력할 것이라고 한다.

—한국 후지쯔

제5회 후지쯔 ASIA 장학생으로 이영환씨 선발

한국후지쯔(주)는 일본 후지쯔가 아시아지역 10개국의 학사학위를 취득한 학생 및 회사원 등을 대상으로 실시하는 「아시아장학금제도」의 한국수혜자로 이영환씨(포항종합제철사원, 29세)를 선발, 지난 6월 27일 장학증서 수여식을 가졌다.

일본 후지쯔의 창립 50주년 기념사업의 일환으로 1985년부터 실시된 이 장학제도는 한국을 비롯하여 아시아지역 10개국 대학원생 및 일반인 중에서 장학생을 선발, 아무런 조건없이 1년간 학비, 생활비, 항공료 등 일체의 비용을 지원한다.

이번에 선발된 이영환씨는 9월 중순경 출국하여 하와이의 일·미 경영과학연구소(JAIMS)에서 5개월, JAIMS 일본지사에서 4개월, 하와이의 Chaminade 대학에서 3개월 등 총 1년간 일본 비지니스와 경제, 국제 비지니스와 사회·문화 등의 교육을 받게 된다.

—한국증권전산

증권 ARS 공동망 구축

그동안 참여업체간의 회선확보에 관한 대립으로 난항을 겪어온 증권 ARS 공동망 구축계획이 한국증권전산(주)와 관련 증권사들의 ARS망 구축으로 시험가동에 들어갔다.

전담기관인 한국증권전산은 최근 50회선을 확보하여 7월중에 시험운영하고 있으며 9월부터 200회선을 확보하여 본격 서비스에 나설 방침에 있다. 이에 따라 한국증권전산은 공동망 구축에 합의한 21개 증권회사중 현재 독자적으로 운영중인 일부 증권사들도 연내에 점차적으로 공동서비스를 전환키로 하였다.

반면에 고려·대신·동양·쌍용 등 4개 증권사는 공동서비스에 참여하지 않고 독자적으로 서비스를 제공키로 했다.

한국증권전산은 공동 ARS 서비스를 우선 서울지역에 제공하고 점차 지방으로 확대 실시할 것이며, 이와 함께 지속적으로 회선규모를 확대하여 일반 전화가입자의 피해를 줄여나갈 계획이라고 한다.

한국증권전산은 증권정보 공동서비스 실시로 인해 대고객 서비스의 향상, 중소증권사의 설비투자에 따른 비용부담의 절감, 단순조회 업무를 대신함으로써 인력절감 효과를 기대하고 있다.

—(주)마이크로소프트사

한글 MS-DOS4.0 발표

지난 7월 4일 (주)마이크로소프트사는 여의도 63빌딩에서 한글 MS-DOS 4.0과 함께 한글멀티플랜2.1, 한글MS-Window2.1에 대한 발표회를 가졌다.

한글MS-DOS4.0은 한글MS-DOS3.2가 갖고 있는 기능을 보강하여 운용체제(OS)를 설치할 때 메뉴방식의 SELECT명령을 이용하여 사용자가 필요사항을 선택할 수 있도록 CONFIG.SYS 및 AUTOEXEC.BAT 화일을 만들어 준다. 그리고 MS-DOS버전 3.30까지는 32MB 이상의 화일을 지원하지 못했으며, 32MB 이상의 하드디스크를 사용할 경우에는 도스파티션을 이용하여 논리드라이브로 사용하였으나, MS-DOS4.0은 도스파티션을 설정하지 않고 최대 2GB까지의 디스크를 사용할 수 있으며, DOS3.30의 도스에서 만들어진 파티션 및 화일들을 모두 사용할 수 있도록 하여 최대 2048MB까지 사용가능토록 하였다.

그외 열린 화일에 대한 확장 cache 지원 등을 보강하여 화일캐쉬기능이 확장되었으며, 명령어들 중에서 ansi.sys, buffers, graphics, tree, format, fdisk 등의 기능이 보강되었다.

한글멀티플랜3.1은 멀티플랜2.0의 기능에서 마우스 기능을 추가한 것으로 OEM들에게 주로 공급되는 표계산용 소프트웨어이다. 이것은 버전2.0에 비해 기능도 보강되었고, 속도도 향상되었으며, 한번에 8개의 화일을 동시에 이용하여 서식의 변화를 자유롭게 지정할 수 있다.

한글MS-Window2.1은 화면을 그래픽으로 처리하여 유저인터페이스가 뛰어나며, 윈도우는 오버래핑 처리를 하므로 CRT화면에 여러개의 프로그램을 동시에 처리할 수 있다. 또 데스크탑 어플리케이션이 포함되어 있어서 별도의 소프트웨어 없이도 간단한 워드프로세싱 계산, 데이터베이스 기능을 수행할 수 있다.

—현대전자

수출용 PC 국내 시판

현대전자(대표:정몽헌)는 수출용 PC를 국내 실정에 맞게 개발하여 7월부터 시판에 나섰다. 이번에 시판된 기종은 AT기종인 SUPER-286E로 i80286을 채택하였고, 12MHz의 클럭속도와 640KB의 메인메모리, 5.25인치(1.2MB)와 3.5"(1.44MB) FDD가 탑재되어 있다.

현대전자에 따르면 금년중으로 AT급 외에도 XT급과 32비트급의 모델도 시판할 예정이다. XT급은 8088-I을 CPU로 하고 10MHz의 클럭속도와 512KB의 메인메모리, 3.5인치(720KB) FDD가 탑재되어 있으며, 32비트인 SUPER-386C는 CPU가 80386이고 클럭속도가 20MHz, 1.2MB의 메인메모리와 5.25인치(1.2MB) FDD를 탑재하고 있다. PC

서클동정

●지식 정보처리연구회●

여름방학이용 회원교육 실시해

지식 정보처리연구회(Information Process Research)는 빠르고 정확한 정보의 보급을 위해 1986년 9월에 만들어진 항공대학의 컴퓨터 써클이다.

동 써클은 현재 정회원 34명과 명예회원 10명으로 구성되어 있고, 87년 신입생에게 베이직 강좌를 실시한 이래로 매년 신입생들에게 컴퓨터 기초강좌를 실시하고 있다.

정기적으로 매주 목요일과 토요일에 모임을 갖고 데이터 뱅크 구축을 위해 이의 기본이 되는 PC통신과 PASCAL, dBASE운영 분야를 집중적으로 공부하고 있다.

지난 7월 3일부터는 20여명의 회원이 참가하여 도서관리 프로그램과 항공대 소개 프로그램, 항공기 구조 설명 프로그램 등을 개발하였다.

이번에 개발된 프로그램들은 8월 31일부터 열리는 항공제 기간동안 전시될 예정이다.

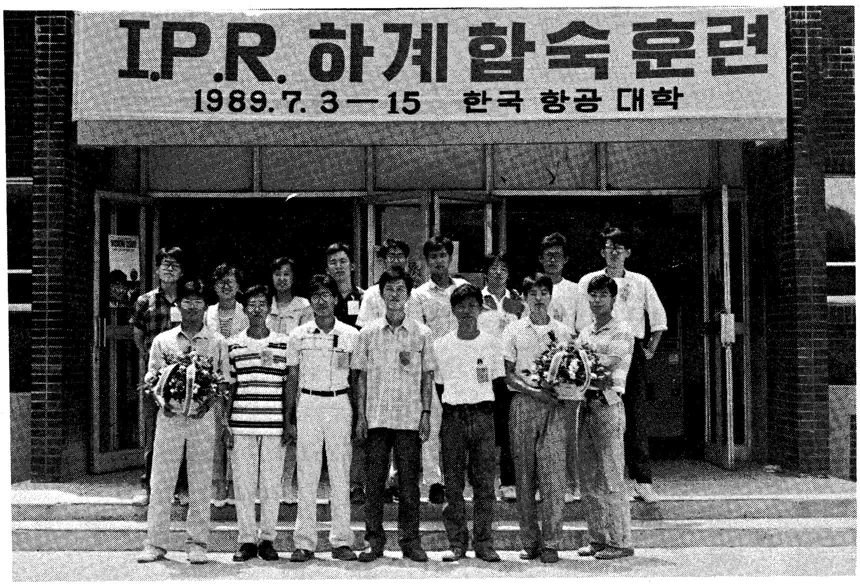
동 써클과 전시회에 대한 사항은 전화 372－5771 항공대 지식 정보처리연구회 회장 이정숙(통신과 2)양에게 문의하면 된다.

●로터스 유저 클럽●

8월12일 정기모임 가져

8월 12일 오후 3시에서 6시까지 경희대학교 경상대 303 강의실에서 정기모임을 갖는다. 이번 모임에는 프린트 그래프와 프린트매크로에 대하여 윤종화씨와 김양완씨가 발표한다.

13일간의 하계합숙훈련을 끝내고 기념으로 한장 찰칵



●온라인회●

컴퓨터 그래픽쇼 준비중

온라인회는 전산병으로 복무하던 사람들끼리 모여 지난 1987년 7월에 결성되었다. 현재의 회원은 12명(회장: 장창호 소프트베이스 근무)으로 구성되어 있고, 연 1회의 정기총회와 격월로 운영되는 임시총회를 통해 각자의 연구분석 자료를 교환하고 있다.

회원의 대다수가 전산 전공자인 특색을 살려 회원 상호간의 친목 도모와 정보의 교환 뿐만 아니라 PC의 대중화 시대에 맞춰 PC의 테크닉 및 기법의 연구를 목적으로 운영되고 있다.

이에 따라 올해안에 PC의 보급이 낙후된 지역을 선정하여 컴퓨터 그래픽 및 간단한 패키지 프로그램을 이용한 컴퓨터 쇼를 개최할 예정이다.

동 모임에 대한 문의 사항은 구본생(전화 552－7921)씨에게 하면 된다.

●온라인 BBS●

5명의 시삽이 독립노드로 운영

개인 BBS로 운영되던 온라인 BBS가 7월 1일부터 인원을 보강하여 5명의 시삽이 각자의 특징을 살려 독립적으로 운영하는 독립 노드의 형태로 전환하였다. 각자가 맡고 있는 분야는 독립 시삽에 의해 운영되지만 게시판 관리와 같은 그밖의 운영상의 중요 부분은 공동으로 운영되고 있다.

각 시삽의 담당분야와 연락처, 운영시간은 다음과 같다.

• 온라인 BBS #1 손희권 PC통신
603－1835 매일 21：00~06：00

•온라인 BBS #2 구준회 SYSTEM S/W 453－4491 매일 22:00～06:00
•온라인 BBS #3 정부영 LANGUAGE 968－0443 매일 18：00～06：00
•온라인 BBS #4 김종민 O.A. S/W 974－0469 매일 23：00～06：00
•온라인 BBS #5 전용상 PACKAGE 888－7145 매일 23：00～08：00

통신방식은 1200/8/N/1 Bell방식이고, 한글은 조합형(NKP) 한글 사용을 원칙으로 한다.

온라인 BBS의 #1인 손희권씨는 온라인 BBS에서 사용하는 전화번호가 컴퓨터 전용회선이 아닌 일반 개인 전화번호이므로 이용자 여러분은 사용시간을 정확히 지켜주길 바란다고 부탁하고 있다.

●KIPC●

사내 전시회 개최 예정

KIPC는 한국전산 사내의 컴퓨터 클럽이다. 이 모임은 PC를 이용하는데 있어 보탬이 되는 소프트웨어에 대한 각종 정보의 교환과 이를 토대로한 회원 상호간의 친목 증진을 위해 1989년 2월에 설립되었다. 현재 회원은 25명으로 회사내 각 부서별로 고른 인원이 분포되어 있으며, 매월 둘째주와 네째주 목요일에 정기적인 모임을 갖고 있다.

회원을 세그룹으로 나누어 TURBO-C, RM-COBOL, MACRO ASSEMBLER, BASIC, MS-COBOL, PASCAL을 중심으로 학습하고 있다. 구성된 회원층도 컴퓨터 사용경력 1년 미만의 초보자부터 10년 이상의 경력을 가진 회원까지 계층이 다양하여 하나의 문제를 해결하는데 있어 다양하게 의견이 수렴되어 쉽게 어려움을 이겨 나가고 있다. 또한 이 모임을 통해 얻은 지식과 경험을 맡은 업무에도 활용을 하고 있어 회사에도 많은 기여를 하고 있다.

연말에는 그동안의 활동에 대한 평가와 정리의 의미로 사내 전시회를 개최할 예정이고, SEK'90 전시회에 참가할 구체적인 프로젝트의 개발을 추진하고 있다. KIPC에 대한 문의 사항은 한국전산(주) 전화 762－1091 최병련씨에게 하면 된다.

●한겨레 BBS●

누구나 대화의 광장에 참여 바라

한겨레 BBS는 유병우(서울대 경제학과 3학년)씨가 운영하는 개인 BBS이다. IBM-XT로 운영되며 24시간 개방한다. 운영 프로그램은 와일드캣 1.03이고, 통신방식은 1200/8/N/1이다.

컴퓨터와 통신분야 뿐만 아니라 모든 분야의 사람들과의 교류를 목적으로 BBS를 개설한 만큼 다양한 주제로 많은 사람들과의 정보 교환 및 시사 토론 등의 기회를 갖는 장으로 이용해주기 바라고 있다. 한겨레 BBS의 전화번호는 (02)849－7861이다.

●전국대학 전산학과 연합회●

제2회 전시회 개최

전국 대학 전산학과 연합회(NCA)에서는 잠실 롯데 월드 7층 특별행사장에서 제2회 전국 대학생 소프트웨어 전시회를 개최한다. 전시기간은 8월 23일부터 8월 28일까지이며 전시 작품은 16비트 퍼스널컴퓨터로 대학생들이 작성한 순수 개인 창작 프로그램들이 전시될 예정이다.

주로 업무용, 교육용, 생활정보, 유틸리티 기타 분야로 나누어 폭넓게 전시될 예정이다.

이번 전시회의 목적은 이를 통한 대학생들의 프로그램 창작수준의 향상과 실제 생활에 이용할 수 있는 작품의 전시로 일반인들에 대한 컴퓨터 마인드의 확산에 있다.

전시회에 대한 자세한 문의사항은 전국대학 전산학과 연합회 사무실 전화 791－1285로 문의하면 된다.

●유니코사●

제4회 전시회 개최

전국 대학 컴퓨터써클 연합회(UNICOSA)에서는 8월 4일부터 8월 7일까지 제4회 UNICOSA전시회를 삼성동 한국 종합전시장(KOEX)에서 개최한다.

이번 전시회는 참가 자격이 대학생 뿐만 아니라 일반인에게도 폭넓게 개방되었고, 전시회기간 동안 컴퓨터에 대한 강연회 등을 개최할 예정이다.

전시회는 크게 하드웨어 제작품과 교육용, 게임용, 비지니스용, 유틸리티용, 연구개발 부문의 소프트웨어 작품들을 전시할 예정이다.

전시회에 대한 자세한 사항은 유니코사 사무실 전화 791－1285로 문의하면 된다.

●마즈다 BBS●

전문적인 지식 교류바라

백용흠(서강대 전자과 4학년)씨는 매일 새벽 1시부터 5시까지 마즈다 BBS를 운영하고 있다.

기존의 메일 이용자들이 같은 분야에 치우쳐 있고, 장난스런 메일도 적지 않으므로 앞으로는 진지한 태도를 가지고 컴퓨터에 대한 광범위하고 전문적인 지식의 교류와 이용자들 사이의 주체의식을 가진 대화가 다양한 분야로 확대되어 나아가길 바라고 있다.

마즈다 BBS의 운영프로그램은 IBM-XT에 RBBS 17.1C로 운영되고 있고, 전화번호는 715－5339이다. PC

국외소식

IBM사
1990년대 핵심이 될「OfficeVision 구상」발표

미 IBM사는 1990년대 동사 컴퓨터 전략의 핵심이 될「OfficeVision 구상」에 대한 윤곽을 분명히 밝혔다.

이 구상은 사무실에서의 컴퓨터에 대한 어플리케이션 환경을 통합화하는 것으로서 3년전부터 동사가 개발하여 온 SAA(Systems Application Architecture) 에 의거한 것이다. 이것은 초심자도 쉽게 조작할 수 있도록 하는 것이 주포인트이다.

OfficeVision 구상에는 동사의 4개 오퍼레이팅 시스템인 VM, MVS, OS/400, OS/2를 포함하고 있다. 대상이 되는 어플리케이션은 도큐멘트프로세싱, 화일링, 메일, 의지결정툴 그리고 그룹스케줄링툴 등이다. 이 구상에 의거한 어플리케이션의 유저인터페이스는 공통의 그래픽환경인 CUA(Common User Access)를 제공하는 것으로 되어 있어서 오퍼레이팅시스템이 달라도 동일한 조작방법으로 어플리케이션을 사용할 수 있다.

이에 따라 OS/2와 PC-DOS판의 어플리케이션은 금년 9월에 제1버전을 발표할 예정이다. 이 버전에서는 LAN을 통해 퍼스널컴퓨터와 그것보다 큰 시스템간에 데이터의 송수신이 가능하게 된다. OS/2판의 경우 가격이 750달러이고, PC-DOS판은 210달러이다.

풀기능을 만족하는 버전은 1990년 제1/4분기 이후에 발표할 예정이며, PC-DOS 터미널에서 액세스할 수 있는 MVS판도 금년 9월중으로 발표할 예정이다. 그리고 OS/2에서 MVS에의 액세스판은 내년 중반정도에 판매할 예정이다. 이 OS/2 버전의 경우는 OS/2에서 메인프레임에의 직접 액세스가 가능할 것이라고 하지만 아직까지 상세한 것은 불명확하다. MVS판의 가격은 15000~50000달러이고 이에 대한 유저의 연간 보수계약료는 2600~9000달러이다.

또 OS/2나 PC-DOS머신에서 System370 에 직접 액세스할 수 있는 VM판의 발표는 1990년 이후가 될 예정인데 이의 가격은 8500~50,000 달러이며, 연간 보수계약료는 1500~9000 달러이다.

OS/400판도 1990년 제2/4분기 이후에 발표될 예정으로 가격은 2900~29000 달러이고 연간 보수계약료는 1900~19000달러로 되어 있다.

이번 발표로 동사의 앞으로 계획이 분명해졌으나 이로 인해 약간의 혼란을 야기하고 있는 것은 사실이다. SAA의 지향하는 바는 유저액세스의 통합·공통화라는 점에 있지만 개발된 배경이나 사용환경도 다른 퍼스널컴퓨터와 메인프레임을 동열로 취급하기에는 아직은 무리가 있는 것 같다.

그러나 OfficeVision에 대한 동사의 독자적으로 거대한 세계를 구축하고자 하는 자세는 SNA에 필적할 것으로 보고 있으나 SNA에 비해 그 지향하는 바가 불명확하다는 의견이 많다.

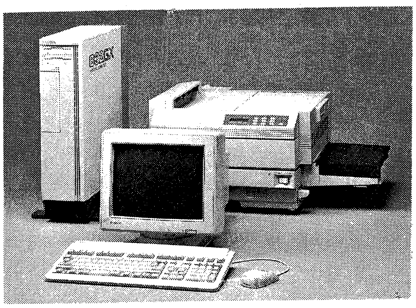
따라서 이번 구상이 침제상태에 빠진 OS/2의 판매확장을 꾀하기 위한 것이라면 이번 발표는 새로운 혼란의 불씨가 될 가능성이 높다.

한 관계자는 이를 마치 "메인프레임의 세계와 같다"고 평하고 있지만 6~8M바이트를 필요로 하는 소프트웨어나 환경에 비해 도가 지나칠 정도의 가격설정을 한 것으로 보아 이 구상은 메인프레임의 세계에서 내려온 아이디어인 것이 분명하다.

일본 히다찌
80386을 탑재한 PC판매

히다찌제작소(주)는 80386을 탑재한 플로어스탠드(Floor Stand)형 퍼스널컴퓨터「B32GX」시리즈 및 주변기기, 소프트웨어를 개발, 판매하기 시작했다.

동 PC는 클럭주파수 20MHz의 80386을 탑재하여 용량 40M바이트의 HDD를 내장한 모델과 클럭주파수 25MHz의 80386을 탑재하여 용량80M바이트



의 HDD를 내장한 모델 등 2모델로 구성되어 있고 이들은 각각 80387을 탑재하는 것도 가능하다.

주요 특징은 다음과 같다.

① 메인메모리는 2M바이트를 표준장착하여 최대16M바이트까지 확장가능
② 용량 32K바이트의 캐시메모리를 탑재하여 메모리액세스 0웨이트를 실현
③ 확장슬롯 4개와 메모리증설 전용슬롯 2개를 장착
④ 인테페이스는 SCSI, RS-232C, 센트로닉스 등을 장착
⑤ 화상해상도는 1120×750도트와 640×400도트의 전환이 가능하며, 4096색 중 16색의 동시표시가 가능
⑥ B16 시리즈용의 소프트웨어, 주변기기를 사용하는 것이 가능

미국 드래곤 시스템사
3만어의 인식이 가능한 음성인식 시스템 개발

미국 드래콘 시스템사(Dragon Systems Inc.)는 3만어를 인식하는 자동청취시스템 「Dragon Dictate 30K」를 개발하였다.

동 시스템은 특정대화자 단어음성인식시스템이며 종래의 동사제품에 비해서 어휘수가 6배 향상하였다. 시스템은 호스트의 퍼스컴에 내장하는 전용포트와 인식소프트웨어로 구성되어 있다. 전용포트는 DSP(TMS32010)를 탑재한다.

인식속도는 15~60워드/분인데 이 속도는 인간이 테이프에 녹음하는 경우에 비하면 늦지만 인간은 음성체크를 위해 테이프의 리와인드를 자주하므로 결국 속도가 비슷해진다고 한다. 호스트는 최저라도 6M바이트의 확장 RAM과 자기디스크가 필요하다. 동사는 미국카네기 메론대학이 DARPA(미국방성 고등연구계획국)의 자금으로 연구 중인 음성인식프로젝트에 참가하고 있다.

NeXT사
NeXT머신용 O.S지원 미비해

NeXT머신이 발표되고 벌써 6개월이 경과하였지만 아직까지도 본격적으로 유저들에게 제품이 출하되고 있지 않다. 북미지역 컴퓨터 딜러인 비즈니스랜드가 일반 유저에 대한 판매계획을 발표하였지만 가장 중요한 시스템의 출하가 지연되고 있어 일반 유저들의 불평이 대단하다.

지난 5월에 OS Ver 0.9가 약 1천명이 넘는 NeXT유저에게 배포되었었다. 이 버전에서는 Ver0.8에서 미완성 부분이었던 오브젝트 C컴파일러가 일단 완성되었다. 또 이 머신의 핵심인 Interface Buider가 대폭으로 개선되었으며 Command line Interface가 새로운 기능으로서 부가되었다.

또한 매킨토시의 콘트롤판넬과 유사한 Preferences도 부가되었다. 이것에 의해서 유저는 키의 반복속도나 마우스의 이동속도, 시각 등을 자유로 조정할 수 있다. 그 외에 추가된 기능은
① 표준의 프린트다이얼로그「Standard Panel」
② 네트워크관리툴 「NetInfo」
③ Apple사의 LaserWriter 대응툴
④ 완성판의 DSP 서포트툴
등을 들 수 있다.

이와 같이 Ver0.9로 된후 많은 기능과 유틸리티, 어플리케이션이 개선되었지만 아직도 완전히 보완되지는 않은 듯 하다.

특히 어플리케이션 기동의 늦음은 처음 시스템이 발표되었을 때부터 문제점으로 지적되었었다. 광디스크가 늦다고 하기 보다도 시스템자체에 문제가 있다고 보는 견해가 대부분이다. 시스템을 광디스크에서 하드디스크에 카피하여 기동하여도 Xwindow를 기동한 Mac이나 Sun3에 비해서 상당히 늦다. Mac이나 Sun3가 NeXT머신보다 늦은 CPU를 사용하고 있고 더구나 메모리가 작은 점을 생각하면 NeXT머신의 OS는 더욱 조정할 필요가 있다고 보인다.

그러나 NeXT머신에 대한 전체적인 평가는 상당히 좋다. 개발측과 머신의 오너인 교육관계자와의 커뮤니케이션도 잘되고 있다고 전해지고 있는데 이러한 양자의 양호한 관계는 머신의 주변을 정비하여 가는데에 절대조건이라고도 할 수 있기 때문에 좋은 현상으로 평가되고 있다. 그리고 동사는 여러가지 문제점들을 개선하여 곧 1.0을 발표할 계획이다.

일본 입석전기
초소형 모뎀 4기종 개발

입석전기(주)는 8㎜ 비디오카세트와 거의 같은 크기의 초소형모뎀 등 4기종을 개발, 판매에 들어갔다. 이번에 발표된 4기종은 다음과 같다.

① 본체사이즈 102(W)×67(D)×21



(H)mm를 실현한 통신속도 1200bps의 전2중모뎀 「MD12FS」
② 보통사이즈의 통신속도 1200bps 전2중모뎀 「MD1200AⅢ」
③ 보통사이즈의 통신속도 4800bps의 반2중모뎀 「MD48HS」
④ 보통사이즈의 통신속도 2400bps의 반2중모뎀 「MD24HS」

특히 MD12FS의 주된 규격은
① AC전원 및 9V 전지에서의 구동이 가능
② AA형의 NCU을 내장
③ 통신규격은 CCITT V.22 / 21 및 Bell 212A / 103을 서포트
④ 코맨드는 Hayes사의 AT코맨드를 지원
⑤ 컴퓨터를 통신상태로 하는 것이며 모뎀의 전원을 자동적으로 ON / OFF 하는 것이 가능

애플사
퍼스컴판매 IBM 제치고 1위 부상

미국 데이터퀘스트사는 지난 5월, 1988년의 미국내 퍼스컴출하대수의 조사결과를 발표하였다.

이에 따르면 IBM사는 전년비 21% 감소인 122만9천대로 크게 부진했으며, 반면 애플사는 전년비 2.5%증가인 127만2천대에 머물렀지만 IBM사를 제치고 처음으로 1위에 서는 결과를 낳았다고 밝혔다.

데이터퀘스트사의 Bill Lempesis씨는 이러한 원인으로 마이크로채널의 비활성화를 들면서 "현재 마이크로채널로 사용할 수 있는 제품은 거의 없다. 이 때문에 유저가 종래의 PC AT의 자산을 이용할 수 있는 호환기로 흘러 버렸다.

그러나 올해는 마이크로채널용 제품의 증가나 IBM의 선전활동의 결과 IBM의 출하대수는 회복될 것이다"라고 분석하였다.

일본 아스키
Intel제품과 호환가능한 수치연산 코프로세서 판매

일(주)아스키는 미국 Integrate Information Technology사가 개발한 수치연산 코프로세서 「2C87」(클럭주파수 10 / 20MHz) 및 「3C87」(클럭주파수 20 / 25MHz)의 샘플출하를 개시하였다.

2C87은 80287과 3C87은 80387과 완전호환을 가진 수치연산 코프로세서이다. 핀배치에서도 호환을 가지고 있으며 80287 / 80387과 바꾸어 꽂을 수 있을 뿐 아니라 2~3배의 처리속도 향상이 가능하다.

주요 특징은
① 각각 오브젝트코드레벨에서의 상위호환을 실현
② IEEE-1985표준포맷에 대응
③ 32 / 64 / 80비트의 부동소수점연산
④ 32개의 어큐뮬레이터스택을 장착하여 4×4행렬 연산이 가능
⑤ 패키지는 2C87이 40핀DIP, 3C87이 68핀 PGA를 채용

등이다.

미국 HP사외 11사
어플리케이션 환경구축을 위한 OMG설립

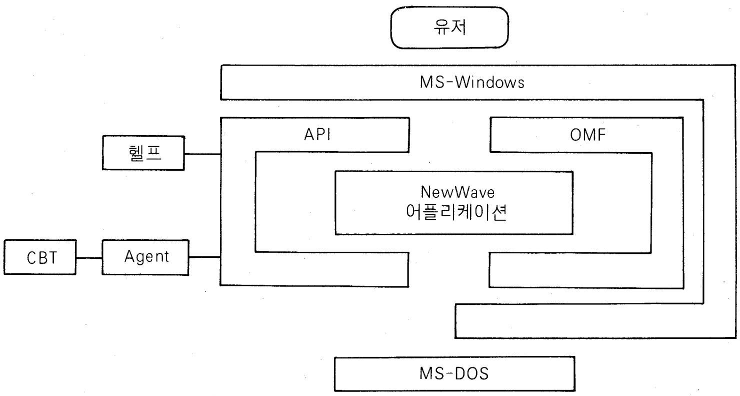
미국Hewlett-Packard Co.(HP) 비롯한 10개사는 오브젝트지향의 어플리케이션환경구축을 목표로 한 OMG-(Object Management Group)를 지난 4월에 설립하였다.

OMG가 목표로 하는 것은 한마디로 말하면 네트워크로 연결하는 컴퓨터간에 퍼진 오브젝트지향의 어플리케이션환경이다.

HP NewWave는 퍼스컴(MS-DOS / Windows)상에서 오브젝트지향의 어플리케이션환경을 만들어내는 소프트웨어로서 현재 미국HP가 OMG에 속한 개발자쪽으로 출하하고 있다.

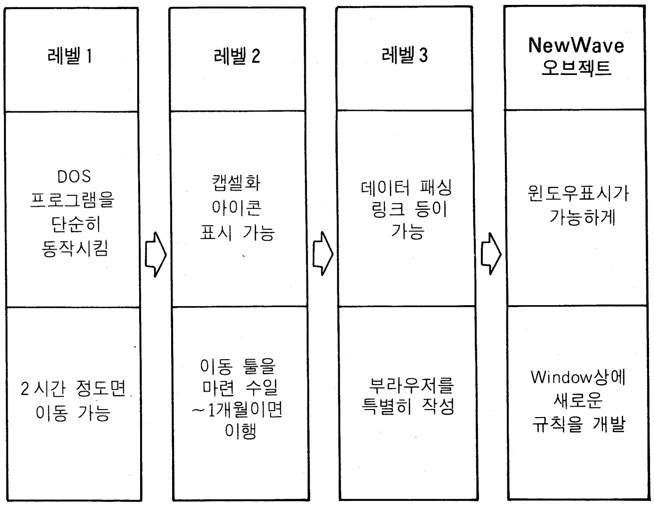
11사의 멤버는 HP, 캐논, Data General Corp., Gold Hill, Prime Computer, Inc., Sun Micro Systems, Inc., Unisys Corp., 3Com, Corpo., American Airlines, Philips International B.V., Soft Switch, Inc., 등이다. 이중 추진역을 맡고 있는 곳은 Data General사이다.

〔그림 1〕 DOS판 NewWave 구성



API : Application Program Interface
CBT : Computer Based Training
OMF : Object Management Facility

〔그림 2〕 기존 DOS 어플리케이션을 NewWave상에 이행한다.



OMG는 NewWave중 일부분(OMF)만을 채용하였는데 미국Apple Computer Inc.는 NewWave의 유저 인터페이스부(MS-Windows)가 동사의 퍼스컴 매킨토시와 매우 유사하다면서 HP와 Microsoft 양사를 제소하여 현재 맞서고 있는 중이다.

OEM 구상의 기본이 되는 HP New-의 구성을 그림 1에 나타냈다. 크게 나누어 오브젝트의 관리를 하는 OMF와 프로그래밍 인터페이스 및 시스템 서비스(헬프기능이나 일종의 매크로인 Agent)를 제공하는 API로 나뉜다.

퍼스컴 세계에서는 무엇보다도 어플리케이션 관련부분을 얼마만큼 갖추었느냐에 따라 그 성능이 좌우된다. 이 때문에 NewWave상에서 DOS상의 기존의 어플리케이션을 살릴 수 있도록 통상의(Windows 어플리케이션은 아니다) DOS어플리케이션에서 NewWave 어플리케이션으로의 이행을 할 수 있도록 하였다. 이 때 그림 2에 나타낸 바와 같이 걸리는 시간과 사용할 수 있는 기능에 따라 3가지의 레벨이 있다.

OMG가 채용을 결정한 NewWave의 OMF는 다음과 같은 역할을 한다.

① 어플리케이션/데이터의 바인딩

예를 들면 「Report」라고 하는 제명의 문서를 편집 중인 워드프로세서를 하나의 오브젝트로 간주한다.

② 오브젝트간의 링크

이 기능을 사용하면 어떤 오브젝트가 유지하는 데이터를 여러개의 오브젝트에서 액세스할 수 있다. 「포토 링크」라고 부르는 기능에 의해 데이터를 새로 바꿀 수 있으면 관련하는 오브젝트도 자동적으로 새로 바뀐다.

③ 오브젝트의 통합

여러개의 오브젝트에의 링크를 가지는 컴파일형 오브젝트를 만들 수 있다.

/Apple사/

새로운 OS「System 7.0」 내년 출하

미국Apple사는 금년 5월9일 미국캘리포니아주 산호세에서 개최한 Worldwide Developers' Conference에서 매킨토시의 새로운 OS「System 7.0」의 규격을 공개하였다.

새로운 OS에서는 가상기억과 프로세스간통신을 서포트하는 것외에

① 단말에뮬레이터나 화일전송 등의 통신소프트웨어와의 인터페이스 커뮤니케이션 툴박스

② 미국Network Innovations사의 CL/1를 기본으로 한 「데이터베이스 액세스 매니저」

③ 아웃라인 폰트를 표시하는 「레이아웃 매니저」를 마련하고 있다.

이 중 ①에 대해서는 Hayse AT 코맨드기능이나 DEC VT100/VT320 단말에뮬레이터, XMODEM 화일전송 등의 통신소프트웨어를 새로운OS 출하시에 마련할 예정이다.

새로운 OS는 2M바이트의 주기억용량이 있으면 매킨토시 Plus에서도 동작할 뿐 아니라 기존어플리케이션의 실행도 가능하다. 가상기억은 68030프로세서 또는 68020프로세서와 68551 메모리관리유니트의 조합을 전제로 하기때문에 매킨토시 Plus/SE에서는 가상기억을 이용할 수 없다. 출하개시는 1990년으로 계획하고 있고 연내에는 동사의 개발자쪽 서비스기관(APDA)을 통해서 기판을 배포할 예정이다.

일본 시스템하우스

카트리지식 HDD 판매

일본시스템하우스는 5.25인치의 자기미디어를 카트리지화한 제거가 가능한 HDD「CADDLE」를 발표하였다.

주된 특징은

① 평균액세스시간은 25ms를 실현

② 1매의 카트리지로 용량 44M바이트를 실현

③ 인터페이스는 SCSI를 채용

④ 카트리지에는 라이트 프로텍트기구를 가짐

⑤ 본체사이즈는 110(W)×350(D)×180(H)㎜, 카트리지사이즈는 138(W)×138(D)×13(H)㎜ 등이다.

적용가능 기종은 PC/AT, AX머신이다.

메모리를 768KB로 확장 사용할 수 있게한

128KB의 비사용 메모리를 사용할 수 있게한 새로운 방안이 발표되어 화제를 모으고 있다. 이 방식을 채택하면 기존의 640KB에 128KB의 메모리를 추가할 수 있어 총 768KB의 메모리를 사용할 수 있으며 램디스크 등 다양한 용도로 메모리를 활용할 수 있다.
따라서 이번에 발표된 메모리 확장방식은 앞으로 메모리의 효율적인 활용방안을 위한 하나의 방법이 될 수 있을 것이다.

메모리의 한계성

우리 주변에서 많이 사용하고 있는 IBM PC/XT 또는 호환기종에서 사용 가능한 메모리는 640KB로 제한되어 있다. 이것은DOS가 지원하고 있는 제어영역이 한정되어 있기 때문이다. 또한 마이크로프로세서인 CPU의 어드레스 버스가 20개인 관계로 최대로 지정할 수 있는 메모리영역은 1MB 이다. 그러나 1MB바이트의 메모리영역도 **그림 1**에서 보는 바와같이 최상위영역 256KB(실제는 더 작음)는 ROM BIOS용으로, 그 아래 128KB는 비디오램 영역으로 사용되고, 나머지 640KB는 시스템관리와 사용자 영역으로 할당되어 있으나 이 영역에서도 사용자가 사용할 수 있는 메모리가 한정되어 있어 실제 사용자영역은 640KB이하로 줄어들게 된다.

그러나 이러한 제한은 AT 기종을 제외하고 어느 시스템에서나공통적으로 가지고 있는 문제점이다.

가상메모리의 이용

가상메모리는 MS-DOS 버전 2.xx이상에서 지원되는 VDISK. SYS를 사용하여 640KB 메모리중 일부를 떼어내어 별도의 메모리를 구성하여 사용하는 것으로서 보조기억장치를 하나 더 추가한 것과 같은 효과를 얻을 수 있는 확장방법으로 주기억장치의 속도만큼 빠르게 사용할 수 있다는 장점이 있으나 주기억장치의 용량을 분할해서 사용하기 때문에 주기억장치의 용량이 감소하여, 줄어든 주기억장치의 크기보다 큰 프로그램은 주기억장치에 적재시킬 수 없다는 단점이 있어 사용시 주의를 필요로 하고 있다.

기존의 메모리 구성을 보면 256K비트와 64비트짜리 램을 같이 사용하여 DOS가 지원하는 640KB만을 구성해 왔으나 64K비트 램 생산의 감소와 256K비트 램 사용의 증가로 64K비트보다 256비트 램 사용이 더 경제적이 되었다. 뿐만 아니라 이 256K비트 램만을 사용하여 메모리를 구성할 경우 전체 메모리는 768KB가 된다. 이 용량은 지금까지 사용가능한640KB에 비해 128KB가 더 많지만 640KB이상의 메모리는 별도의 운용 소프트웨어가 없으면 사용할수 없어 지금까지는 사용하지 않은채 방치되어, 결과적으로 메모리의 낭비를 초래했다고 볼 수 있다.

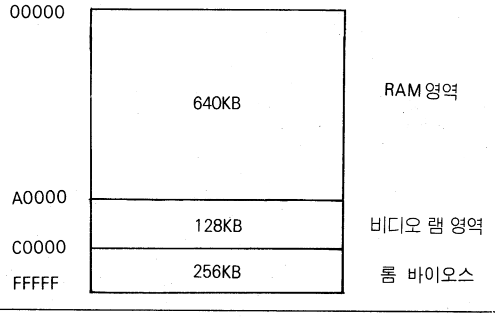
128KB의 이용방법

보급형 PC가 많이 나오면서 메모리 용량에 대한 문제가 상당히 대두되고 있는 요즈음 희망전자에서는 이 128KB의 비사용 메모리를 사용할 수 있는 방안을 발표하였다. 이 방법은 기존의 640KB의 메모리는 그대로 사용하고, 그 동안 사용하지 않았던 128KB 영역을 하드웨어의 재구성과 소프트웨어의 제공으로 추가적인 보조기억장치로서 사용하도록 한 것이다.

이번에 개발된 128KB 가상 메모리를 이용한 국민보급형 PC의 구성은 256K비트 DRAM을 사용하여 256KB씩 3줄로 구성하여 생기는 768KB메모리중 640KB를 뺀 나머지 128KB를 가상메모리로 이용할 수 있도록 하였다. 그러나 그림 1에서 보여준것처럼 남는 영역인 128KB가 비디오램 영역(A0000H-BFFFFH)으로 이미 사용되고 있으므로 768KB 영역 중 마지막 256KB의 영역에 Bank 개념을 도입하여 Bank0, Bank1로 분할, **그림 2**에서와 같이 어드레스 80000H에서 9FFFFH 부분의 램영역을 공유하도록 설계되었다.

이 메모리 맵에서 Bank0 영역은 512KB와 합쳐져서 640KB의 메모리를 구성하고, Bank1 영역은 제공되는 HPDISK. SYS를 이용하여 RAM DISK로 사용된다. Bank1 영역에 해당하는 메모리는 현재 DOS상에서 XT 호환기종의 주메모리를 640KB만 지원하고 있으므로 이 확장영역을 주메모리영역이 아닌 RAM디스크와 같은 한정된 용도밖에 활용할 수 없고, 앞으로 DOS가 지원할 수 있는 메모리 영역을 확장한다면 640KB 이상의 용량을 메인메모리로 사용할 수 있을 것이다.

〔그림 1〕 640KB 메모리 맵



HOPE-16000S

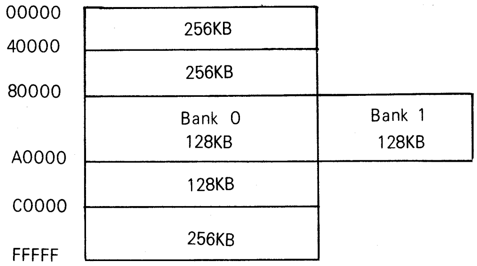
손영주 / 취재부 기자

하드웨어적인 차이

이 가상메모리를 구성하는데 있어서 기존 시스템과의 하드웨어적인 차이점은 어드레스 A19와A18로 세번째 메모리(80000H - BFFFFH)를 선택하는 것은 동일하고, 다시 이 메모리의 영역을 128KB씩 양분하여 어드레스 A17 에 0의 값을 주면 위에서 선택된 메모리 영역 (80000 H-BFFFFH) 중 80000H에서 9FFFFH까지의 영역이 선택되어 비디오 램 영역과 겹치지 않게 고정시켜 놓은 것만이 다를 뿐이다.

그러나 실제 사용된 세번째 램의 메모리 영역(80000H-BFFFFH)에서 어드레스 A17로 선택되어진 영역을 뺀 나머지는 비디오 영역이 따로 존재하고 있으므로 사용자 영역으로 사용하지 못하고 남는 영역이 된다. 이 방법을 구현한 것이 희망전자의 보급형 PC HOPE-16000S 제품이다. 여기서 구성된 256KB의 램 어드레스 중 A17을 Bank 셀렉터에 입력, 남는 메모리 128KB를 Bank1로 지정하여 사용할 수 있는 것이다. 이 뱅크의 선택은 지원되는 프로그램으로 가능한데 뱅크 레지스터(실제 I/O)번지값을 세팅(제품에 따라 다를 수 있음)하고 지정된 데이터라인 (이 제품에선 D0)을 사용하여 이 값이 0일 경우 Bank0, 1일 경우 Bank1의 영역을 선택하게 된다. 이것에 대한 간단한 프로그램을 보면 다음과 같다.

〔그림 2〕 768KB 메모리 맵



뱅크 1의 영역을 사용할 경우

```
MOV   DX,BANK REG.
MOV   AL,Bank1
OUT   DX,AL
```

뱅크 0의 영역으로 돌아올 경우

```
MOV   DX,BANK REG.
MOV   AL,Bank0
OUT   DX,AL
```

위와 같은 뱅크전환은 시스템이 처음 부팅되면서 검색된 HPDISK. SYS 화일이 사용 여부에 따라 자동으로 전환해 주게 된다. 이와같은 방법은 DOS에서 지원되는 VDISK. SYS을 이용한 가상메모리 동작과 같다. 다만 여기서 다른 점은 기존의 640KB의 메모리를 사용하는 것이 아니고 768KB 로 확장된 메모리중 사용하지 않았던 128KB의 영역을 사용하고있다는 점이다. DOS에서 제공되는 RAMDISK. SYS 또는 VDISK. SYS로 가상메모리를 구성할경우 사용자의 영역을 감소시켜 응용프로그램 운영상 메모리 부족현상을 불러일으킬 수 있지만 HPDISK. SYS로 Bank1의 128KB 램영역을 사용할 경우 아직은 128KB에 한정되어 있지만 사용자가 사용하는 640KB의 영역은 줄어 들지 않는다.

이상과 같은 방법은 실제 사용하고 있는 XT급 호환기종에서도 가능하나 시스템 자체의 하드웨어적인 변경이 요구되고 별도의 운용 프로그램이 있어야 하므로 기존의 유저들이 이용하기에는 다소의 시간이 걸릴 것으로 보인다. 하지만 이번에 발표된 방법은 컴퓨터 사용자들에게 많은 관심을 불러일으킬 수 있을 것이다.

회로등의 추가로 더 많은 메모리를 확장할 수도 있는데, 현재로서는 지원되고 있는 DOS의 제한으로 인해 확장카드가 갖는 램디스크의 역할밖에는 수행할 수 없을 것이다. PC

한글이 깨어지는 것을 방지해주는 HGSPLIT.PRG

김기남 / 성균관대학교 수학과

dBASE에서 리포트를 작성하려고 할 경우, 다단 리포트(한 레코드를 여러줄에 인쇄하는 경우)는 프로그램을 작성하여 제어해야 하는데 한 필드가 문자필드로서 너무 큰 경우에는 문자열 분리를 하여 두라인에 걸쳐서 표시하여 주는 경우가 있다. 이때 영문의 경우는 별 문제가 되지 않지만 한글은 한글자가 2바이트로 이루어져 있기 때문에 첫번째 바이트에서 분리가 이루어지면 한글이 깨져 알아볼 수 없게 된다. 이것이 dBASE의 SUBSTR() 함수를 사용할 수 없는 이유이다.

이제 소개할 프로시듀어(procedure)는 이러한 한글로 이루어진 문자열을 특정한 길이만큼 분리하여 주프로그램에 다시 돌려주게 된다. 필자는 프로그램 작성중에 이러한 문제에 부딪히게 되었는데 처음에는 한글코드의 식별이 어려울 것 같아 무척 걱정했는데 막상 작성하다 보니 주석을 빼면 21라인 밖에 안되는 간단한 프로시듀어가 되었다.

문자열 분리의 알고리즘

한글은 2바이트로 되어 있으며 조합형은 MSB(Most Significant Bit)를 1로 세팅하기 때문에 IBM 그래픽 캐릭터처럼 보이게 된다. 즉 아스키코드 값이 125를 넘게 된다. 반면에 7비트 완성형 한글의 대표적인 경우라 할 수 있는 청계천 한글은 그래픽문자로는 보이지 않지만 소문자와 대문자의 조합이나, 특수문자와 영문자의 조합이 한글로 보이게 된다. 다행히도 이들 한글은 소트를 위하여 첫번째와 두번째 바이트가 일정한 범위를 가지고 있는데 이것이 한글 문자열 분리의 열쇠라 할 수 있다.

원하는 포인터의 아스키코드 값을 판별해서 이 아스키코드값이 한글 첫번째 바이트가 아니면 분리할 수 있다. 두번째 바이트와 첫번째 바이트에 공통적인 아스키코드 문자도 있으므로 조건에 맞지 않으면 계속 포인터를 앞쪽으로 이동시키게 되는데 최악의 경우 이동된 포인터가 목적하던 포인터보다 훨씬 앞쪽 바이트에 위치하게 될 것이다.

이를 보상하는 방법은 의외로 간단한데 분리하려는 목적포인터에서 분리했을 경우 한글이 깨질 확률은 1/2이 된다. 이를 이용하기 위하여 목적포인터에서 현재의 포인터를 빼서 몇 바이트의 오차가 있는지 산출한 후 뒤의 문자는 모두가 2바이트의 한글로 되어 있다고 가정하면 오차값을 2로 나누어서 나머지가 0이 되면 목적포인트에서 분리를 해도 되고 그렇지 못하다면 1바이트 앞에서 분리를 시도한다.

〔리스트 1〕 HGSPLIT.PRG

```
*
*****************  HANGUL STRING  SEPARATER   ****************
*
*                                    June. 1989.    For The Logos.
*
* PROCEDURE HGSPLIT
*
*      source : SOURCE STRING           -> 분리 되어질 문자열
*      first  : FIRST  PART STRING      -> 분리된 후 첫번째 문자열
*      second : SECOND PART STRING      -> 분리된 후 두번째 문자열
*      point  : CUT POINTER FOR source  -> 문자열 분리를 위한 기준점
*
PARAMETER source,first,second,point
PRIVATE cut0,cut1,cnt00,cnt01,hch00
cut0  = LEN(TRIM(M->source))
cnt00 = M->point
DO WHILE M->cnt00>0
  hch00 = ASC(SUBSTR(M->source,M->cnt00,1))
************      한글 두번째 바이트의 식별을 위한 조건식      ************
  IF .NOT.(M->hch00>=126.AND.M->hch00<=255)
    EXIT
  ELSE
    cnt00 = M->cnt00 - 1
    LOOP
  ENDIF
ENDDO
*******       분리된 문자열을 최적화 시키기 위한 조건식       ************
IF M->cnt00<(M->point-1)
  cnt01 = M->point - M->cnt00
  cnt00 = IIF(MOD(M->cnt01,2)=0, M->point,(M->point-1))
ENDIF
cut1   = M->cut0 - M->cnt00
****************              문자열 분리              ****************
first  = SUBSTR(M->source, 1,M->cnt00)
second = SUBSTR(M->source,(M->cnt00+1),M->cut1)
RETURN
```

HGSPLIT.PRG의 분석

1~12행 : HGSPLIT.PRG 대한 코멘트(설명)이다. 6행을 주의하여 보기 바란다. 이 예에서는 화일이름이 HGSPLIT.PRG로서 프로시듀어 HGSPLIT가 단독으로 존재한다고 가정하였으므로 6행을 코멘트 처리하였지만 여러개의 다른 프로시듀어와 함께 공존할 경우에는 6행 1열의 코멘트 부호인 '*'를 없애 주어야 한다.

13행 : 주프로그램으로부터 받는 변수를 선언한 부분으로 SOURCE는 분리하기 위한 문자열이고 FIRST, SECOND는 분리된 후 첫번째, 두번째 문자열이며 POINT는 분리를 원하는 포인트(숫자)이다. FIRST, SECOND는 프로시듀어 호출전에 주프로그램에서 문자열변수로 선언하여 주면 되며 이것을 PUBLIC 선언하여 주프로그램으로 넘겨주지 않은 이유는 주프로그램과의 변수충돌을 피하기 위함이다.

14행 : HGSPLIT.PRG에서 사용된 지역 변수를 선언한 부분으로 이렇게 PRIVATE 선언을 하여 줌으로써 주프로그램과의 불필요한 변수 충돌을 피할 수 있다.

15~16행 : CUT0는 분리될 문자열의 실제 길이를 저장하는 변수이고 CNT00은 분리시킬 포인터값을 저장하는 변수이다. 이것은 나중에 POINT 값을 변경시키는 대신 CNT00을 변경하기 위함이다.

17~26행 : 핵심 루틴으로 CNT00을 감소시키면서 조건이 맞을 때까지 루프를 돌게 된다. 이때 조건이 되는 20행이 가장 중요한데 현재는 조합형 한글에 맞추어져 있다. 즉 검출된 1바이트의 아스키코드 값이 그래픽 문자가 아닐 경우에 분리시키기 위해 루프를 벗어나도록 되어 있는데 청계천 완성형의 경우에는

IF .NOT. (M —> hch00>94.AND.M —> hch00<127)

와 같이 조건식을 주면 된다. 이것은 청계천 완성형 한글의 첫번째 1바이트의 아스키코드 범위가 95~126인 것에 준한다.

사실상 조합형의 경우에도 ()안에 한글 첫번째 1바이트 코드값의 범위가 들어가야 하지만 적용 범위의 확장을 위하여 그래픽 코드값으로 대체시켰다. 이것은 일반해로서 작용하여 거의 모든 조합형한글 문자열에 적용될 것이다. 만약 한글이 깨진다면 사용하는 한글카드의 한글 첫번째 바이트의 코드값으로 대체시키기 바란다.

만약 발송용 주소를 인쇄하려고 할 경우는 목적하는 포인터에 근접해서 분리하는 것 보다는 SPACE에 의해서 분리한 후 확대 인쇄하는 것이 더 멋있게 보이는데 이때는 조건식을

IF M —> hch00=32

와 같이 하면 된다. HGSPLIT.PRG의 28~32행은 필요없으므로 삭제하도록 한다. 이 때 두번째 문자열의 길이가 원래의 문자열의 길이와 거의 같아질 수도 있다는 점을 고려해야 한다. 발송용 주소 이외에는 이 조건식을 지양하여 주기 바란다.

29~30행 : 앞에서 언급한 바와 같이 검출된 분리점이 목적한 포인터와 상당한 차이가 있을 때 이를 보완해 주기 위한 루틴으로 분리시킬 위치가 목적한 포인터인가 목적한 포인터 1바이트 앞인가를 식별하기 위한 루틴이다.

34~35행 : 앞의 정보들을 종합하여 문자열을 분리시키는 루틴이다.

HGSPLIT.PRG의 활용

리스트 2는 HGSPLIT.PRG의 활용예를 보인 것이며 화면 1은 실행예를 보인다. mesg에 분리하기 위한 문자열을 저장하고 분리된 문자열을 받기 위하여 string 1, string 2를 42바이트의 문자변수로 HGSPLIT.PRG 호출 이전에 선언하고 있다. 이때 꼭 42바이트일 필요는 없으며 실제로는 변수선언을 해주기만 하면 된다. 분리시킬 포인터는 42로 선정을 하였다. 이때 분리된 두번째 문자열이 42바이트를 넘었기 때문에 다시 두번째 문자열을 mesg에 저장하여 프로시듀어 HGSPLIT.PRG를 다시 호출하고 있다. 만약 dBASE의 프로시듀어도 파스칼이나 C 언어의 함수와 같이 재귀호출(Re-

〔리스트 2〕 HSTEST.PRG

```
*
SET TALK OFF
mesg = "진실로 참되고 옳음이 죽어지고 숨어져야 하는 이 계절에 나의 "+;
       "뜨거운 노래는 여기 언 땅위에 깊이 묻으리라."
STORE SPACE(42) TO string1,string2
DO HGSPLIT WITH MESG, STRING1, STRING2, 42
?
? " ** 첫 번 째   실 험  **"
?
? " ### 분리되기 전의 문자열     -> " + mesg
? " ### 분리된 후 첫번째 문자열 -> " + string1
? " ### 분리된 후 두번째 문자열 -> " + string2
mesg = string2
STORE SPACE(42) TO string1, string2
DO HGSPLIT WITH MESG, STRING1, STRING2, 42
?
?
? " ** 두 번 째   실 험  ** "
?
? " ### 분리되기 전의 문자열     -> " + mesg
? " ### 분리된 후 첫번째 문자열 -> " + string1
? " ### 분리된 후 두번째 문자열 -> " + string2
```

〔화면 1〕 HGSPLIT.PRG의 실행

```
** 첫번째 실험 **

### 분리되기 전의 문자열    -> 진실로 참되고 옳음이 죽어지고 숨어져야 하는
이 계절에 나의 뜨거운 노래는 여기 언 땅위에 깊이 묻으리라.
### 분리된 후 첫번째 문자열 -> 진실로 참되고 옳음이 죽어지고 숨어져야 하
### 분리된 후 두번째 문자열 -> 는 이 계절에 나의 뜨거운 노래는 여기 언 땅위
에 깊이 묻으리라.


** 두번째 실험 **

### 분리되기 전의 문자열    -> 는 이 계절에 나의 뜨거운 노래는 여기 언 땅위
에 깊이 묻으리라.
### 분리된 후 첫번째 문자열 -> 는 이 계절에 나의 뜨거운 노래는 여기 언 땅
### 분리된 후 두번째 문자열 -> 위에 깊이 묻으리라.
```

cursion) 이 가능하다면 훨씬 더 강력해졌겠지만 유감스럽게도 dBASE의 함수는 재귀호출(함수자신을 부르는 경우)을 지원하지 않는다.

참고로 예에서 문자열이 너무 길어진 변수에 대해서는 편집상 임의로 리턴문자를 삽입시켜서 개행을 시켰으므로 이 점 양해를 바란다.

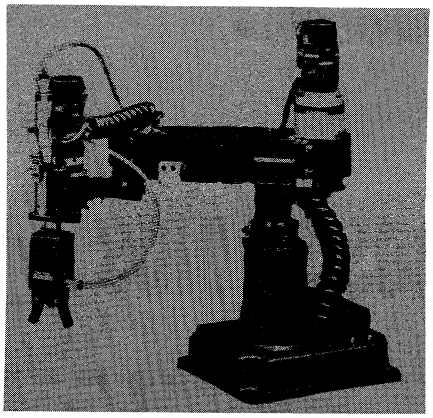
마이크로프로세서를 이용한

FA SYSTEM 과정

(생산자동화)

※ 개강 : 매월 5일

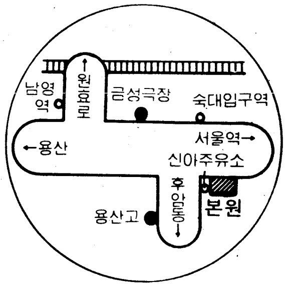
講義案內 (2개월 완성)



본원에서는 다년간 Computer Hardware 과정을 교육하여 많은 수료생을 배출, 각계에서 공헌하고 있는 바 금번 생산업계의 요청에 따라 컴퓨터 프로세서를 이용한 생산체제와 산업용 로보트의 발전으로 제품과 제조공정의 설계 및 계획, 공장 현장에 있는 생산 제조장치의 컴퓨터 제어, 원료 및 자재의 유동제어, 공간 활용 및 작업계획 등의 조직적 기획 등의 필요성에 따른 FA 과정의 이론, 실기를 위주로 특강을 하게 되었으니 관계 직원의 많은 참여 바랍니다.

■ **강의일정**

주별	일자 (과정)	강습내용 이론 및 실기	주별	일자 (과정)	강습내용 이론 및 실기
1주	1일차	마이크로 컴퓨터 시스템	5주	1일차	DC MOTOR 이론 및 특성
	2일차	마이크로 컴퓨터 KKH80A H/W		2일차	DC MOTOR 응용 회로 설계
	3일차	Z80 어셈블리 언어		3일차	STEP MOTOR 이론 및 특성
2주	1일차	마이크로 컴퓨터 KKH80A S/W	6주	1일차	STEP MOTOR 응용 회로 설계
	2일차	PIO 응용 실습		2일차	센서의 종류와 특성
	3일차	Z80 인터럽트 실습		3일차	마이컴에 의한 센서 활용
3주	1일차	CTC 및 DC MOTOR DRIVER 제작	7주	1일차	i8251 및 RS-232C 통신 실험
	2일차	A/D변환, D/A변환 실습		2일차	ROM WRITER의 제작
	3일차	STEP MOTOR DRIVER 제작		3일차	LCD DISPLAY MODULE의 사용법
4주	1일차	모니터 프로그램 설계	8주	1일차	IBM PC 인터페이스 실습
	2일차	응용 프로그램 분석 I		2일차	ONE CHIP 마이크로 컴퓨터
	3일차	응용 프로그램 분석 II		3일차	8748, 8749, 8035, 8039, 8751 응용



전자산업 4반세기를 지켜온 기술인의 전당

공인 **한국전자통신학원** 컴퓨터기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

DOS와 BIOS 인터럽트

박승제 / 여주컴퓨터 학원장

인터럽트는 '가로채기', '끼어들기' 등으로도 번역되는 것으로 컴퓨터가 어떤 작업을 처리하는 과정에서 급박한 사항이 발생하여 현재의 일을 잠깐 멈추고 다른 일을 하는 것을 말한다. 예를 들어 독자들이 책을 보고 있는데 전화 벨이 울리면 보던 책을 잠시 놓고 전화를 받고, 전화가 끝나면 다시 책을 읽어나가게 된다. 여기서의 전화가 인터럽트에 해당한다.

인터럽트는 기계적으로 처리하는 하드웨어 인터럽트와 소프트웨어 인터럽트가 있는데 하드웨어 인터럽트는 기계 내부에서 처리하는 인터럽트를 말하며, 소프트웨어 인터럽트는 프로그램상에서 인터럽트를 수행하는 것을 말한다. 우리가 여지껏 배워왔던 INT 21H 등은 소프트웨어 인터럽트에 해당한다. 우리는 이런 인터럽트가 어떻게 처리되는지는 아직 알 필요는 없고 (좀 더 전문가적인 경지에 이르려면 반드시 알아야겠지만) 그 기능들을 적절히 활용하므로써 보다 효과적인 프로그램을 작성할 수 있다.

이 인터럽트들은 대부분 레지스터에 특정한 값을 기억시킨 다음 호출할 경우 그 기능을 수행한 뒤 필요시에는 연산 결과를 다시 레지스터에 넣어서 돌아오게 된다. 여기서 인터럽트를 호출하기 전에 대입하는 자료와 레지스터들을 '엔트리 파라미터(Entry Parameter)', 되돌아오는 자료와 레지스터들을 '리턴 파라미터(Return Parameter)'라고 부른다. 여기에 게재하는 자료들은 어셈블리 프로그램을 작성하는데 없어서는 안될 아주 유용한 것이다. 단, 여기서 사용하는 인터럽트와 펑션 번호는 모두 16진수이므로 주의가 필요하다.

인터럽트 일람표

• INT 0H : 0에 의한 나눗셈

이 인터럽트는 0으로 나누는 경우나 나눗셈의 몫이 16비트 / 8비트를 벗어난 경우에 호출된다. 즉, 32비트÷16비트=16비트, 16비트÷8비트=8비트의 형식이 아닐 경우를 말한다. (나눗셈의 형식은 89년 5월호의 본 강좌를 참조하기 바란다.) 또한 특별한 경우로 스택이 파괴되었을 경우에도 INT 0이 호출된다. 보통 'Divide Overflow'의 메시지가 출력이 되며 프로그램의 수행은 중단된다.

• INT 1H : 한 단계 실행

이 인터럽트는 디버거의 Trace 모드에서만 이용된다. 명령을 한 스텝 실행한 후 멈추게 된다.

• INT 2H : 패리티 체크

이것은 하드웨어 인터럽트이다. 사용자 메모리의 패리티를 체크하는 부분에 에러가 있을 때 이 인터럽트가 호출된다. 'Parity Check Error'를 호출한다.

• INT 3H : Break Point

디버거에서만 사용한다. 이 인터럽트가 호출되면 그 시점에서 프로그램이 종료되며, 어버이 프로세서로 제어가 넘어간다.

• INT 4H : 오버 플로우

오버플로우 조건이 있을 경우에 이 인터럽트가 호출된다. 보통은 아무일도 하지 않고 그냥 리턴되나 사용자가 처리 루틴을 변경하여 특정한 일을 수행시킬 수 있다.

• INT 5H : 하드 카피

화면을 프린터로 하드카피한다. 키보드에서 〈Shift〉+〈PrtSc〉키를 누를 때 호출된다.

• INT 6H, 7H : 보유된 인터럽트

차후의 확장성을 고려해 현재에는 사용하지 않는다.

• INT 8H : Timer 인터럽트

하드웨어 인터럽트이다. 매초 18.2번 호출되며 내부 시간을 변경하기 위해 사용된다. 이 인터럽트가 호출되면 뒤에 나올 INT 1CH를 자동적으로 Call하게 된다. 타이머 인터럽트를 사용하려는 경우에는 이 인터럽트를 건드리지 말고 INT 1CH를 이용하는 것이 바람직하다.

• INT 9H : 키보드 인터럽트

키보드에서 문자를 입력하면 이 인터럽트가 호출된다. 이 인터럽트는 키스트록(Keystroke)을 읽어서 스캔 코드와 아스키 코드로 분해하여 키버퍼에 저장하게 된다.

• INT 0AH~0FH : 보유된 인터럽트

• INT 10H : 화면 처리

이 인터럽트는 화면 처리에 관한 것으로 16가지의 기능이 있는데 각각의 루틴은 AH로 지정한다.

(0) 화면의 모드를 설정한다.

[Entry] AH : 0, AL : 화면 모드 번호

〈화면 모드〉

0 : 40*25 흑백
1 : 40*25 컬러
2 : 80*25 흑백
3 : 80*25 컬러
4 : 320*200 컬러
5 : 320*200 흑백
6 : 640*200 컬러
7 : 80*25 모노크롬

(1) 커서의 모양을 지정한다.

[Entry] AH : 1, CH : 커서의 시작 줄 번호, CL : 커서의 끝 줄 번호

(2) 커서의 위치를 지정한다.

[Entry] AH : 2, DH : 커서의 수직 위치, DL : 커서의 수평 위치, BH : 페이지 번호

커서의 좌표는 좌측 상단(Home Position)을 (0,0)으로 한다.

(3) 커서의 위치를 찾는다.

[Entry] AH : 3, BH : 페이지 번호
[Return] DH, DL : 커서의 좌표, CH, CL : 커서의 모양

(4) 라이트 펜의 위치를 찾는다.

[Entry] AH : 4, BH : 페이지 번호
[Return] AH : 0 라이트펜이 접촉되지 않았음, AH : 1 라이트 펜이 접촉되었음, DH, DL : 라이트 펜 위치(row, column), CH : 라이트 펜의 수직 위치(픽셀 단위 : 0~199), CL : 라이트 펜의 수평 위치(픽셀 단위 : 0~319, 639)

(5) 화면의 페이지를 바꾼다.

[Entry] AH : 5, AL : 페이지 번호

〈페이지 번호〉

AL : 0~7(스크린 모드 0, 1일 때)
AL : 0~3(스크린 모드 2, 3일 때)

(6) 윈도우를 위로 스크롤한다.

[Entry] AH : 6, AL : 스크롤할 라인 수, CH, CL : 윈도우의 좌측 상단 모서리 좌표, DX, DL : 윈도우의 우측 하단 모서리 좌표, BH : 스크롤된 자리의 배경색의 속성 코드

*스크롤 할 라인 수가 0이면 윈도우 전체를 지운다.

(7) 윈도우를 아래로 스크롤한다.

AH에 7을 넣는다는 것만 다를 뿐 (6)과 동일하다.

(8) 커서 위치의 1문자를 읽는다.

[Entry] AH : 8, BH : 페이지 번호
[Return] AL : 문자의 아스키 코드, AH : 문자의 속성

(9) 커서의 위치에 문자와 속성을 쓴다.

[Entry] AH : 9, AL : 아스키 코드, BH : 페이지 번호, BL : 속성 코드, CX : 출력할 문자 수

(0A) 커서의 위치에 문자만 쓴다.

[Entry] AH : 0A, AL : 아스키 코드, BH : 페이지 번호, CX : 출력할 문자 수

(0B) 파레트의 색을 지정한다.

[Entry] AH : 0B, BH : 전경색(1), 배경색(0)의 지정, BL : 파레트 번호
〈파레트 번호〉 0 : Green / Red / Yellow 1 : Cyan / Magenta / White

(0C) 그래픽 화면에 Dot 출력

[Entry] AH : 0C, AL : 컬러(0~3), DX : 세로 좌표(0~199), CX : 가로 좌표(0~319, 639)

*AL의 최상위 비트가 1이면 현재의 값과 XOR 된다.

(0D) 그래픽 화면에서 Dot 입력

[Entry] AH : 0D, DX : 세로 좌표(0-~199), CX : 가로 좌표(0~319, 639)
[Return] AL : 컬러 코드(0~3)

(0E) 그래픽 화면에 1문자를 쓴다.

[Entry] AH : 0E, AL : 아스키 코드, BL : 전경색 코드

이 루틴에서는 1~31의 제어코드를 인식하지 않고 세미그래픽 문자로 출력한다.

(0F) 화면의 상태를 조사한다.

[Entry] AH : 0F
[Return] AH : 화면의 최대 컬럼 수, AL : 화면 모드, BH : 현재의 페이지 번호 화면 모드는 (0)과 같다.

- **INT 11H** : 장비의 구성 상태를 조사한다.

[Return] AX : 장비의 구성 상태가 비트로 세트된다.

〈비트의 속성〉

0 : 드라이브 존재 유(1) 무(0)
3, 2 : 내장된 ROM의 크기(00=16K, 01=32K, 10=48K, 11=64K)
5, 4 : 그래픽 보드의 종류(01=40*25 컬러 보드, 10=80*25 컬러 보드, 11=80*25 모노크롬)
7, 6 : 드라이브의 갯수(00=1개, 01=2개, 10=3개, 11=4개)
11, 10, 9 : RS-232C 카드 설치시 사용
12 : 조이스틱 포트 설치 여부
15, 14 : 프린터 포트의 수

- **INT 12H** : 메모리의 크기를 조사한다.

[Return] AX : 1K 블록의 갯수(640KB인 경우 640을 나타내는 280H가 리턴된다)

- **INT 13H** : 디스크 처리

이 인터럽트는 디스크에 관한 것으로 모두 20개의 루틴이 있으며 작업중 에러가 발생한 경우에는 캐리 플래그가 세트되며, 에러 코드는 AL로 설정된다. 또한 드라이브의 번호는 DL로 지정하는데 DL이 80H미만이면 플로피 디스크를, 80H 이상이면 하드디스크를 대상으로 작업을 하게 된다.

(0) 디스크를 리세트한다.

[Entry] AH : 0,
[Return] AL : 에러 코드. (에러 코드는 (1)을 참조한다.)

(1) 디스크에 대한 마지막 작업 상태를 읽는다.

[Entry] AH : 1, DL : 드라이브 번호
[Return] AL : 에러 코드

〈에러 코드〉

00 : 에러 없음
01 : 콘트롤러에 대한 명령이 잘못되었다.
02 : 지정한 주소가 발견되지 않는다.
03 : 디스켓에 프로텍트 탭이 붙어 있다.(플로피)
04 : 섹터가 발견되지 않는다.
05 : 디스크의 리세트가 실패했다.
07 : 드라이브에 대한 파라미터가 잘못 지정되었다.
09 : 지정된 메모리가 세그먼트의 끝을 넘어섰다.

0B : 불량 트랙이 발견되었다.
10 : bad error check seen(하드)
11 : Data is error Corrected
20 : 콘트롤러의 에러
40 : 드라이브의 동작이 원만하지 못함.
80 : 디스크에서 신호가 오지 않는다.
BB : 정의되지 않은 에러가 발생했다. (하드)
FF : 드라이브 헤드의 에러(하드)

(2) 섹터를 메모리에 읽어들인다.

[Entry] AH : 2, AL : 읽어들일 섹터 수(플로피=1~8, 하드=1~80), DL : 드라이브 번호, DH : 헤드 번호, CH : 실린더나 헤드 번호, CL : 섹터 번호, BX : 메모리의 선두 번지, ES : 메모리의 세그먼트
[Return] AH : 에러 코드

(3) 메모리 내용을 디스크의 섹터에 쓴다.

[Entry] AH : 3, AL : 쓸 섹터 수(플로피=1~8, 하드=1~80), DL : 드라이브 번호, DH : 헤드 번호, CH : 실린더나 헤드 번호, CL : 섹터 번호, BX : 메모리의 선두 번지, ES : 메모리의 세그먼트
[Return] AH : 에러 코드

(4) 디스크의 섹터를 체크한다. 섹터에 에러가 없으면 내부 버퍼로 읽어들인다.

[Entry] AH : 4, AL : 조사할 섹터 수(플로피=1~8, 하드=1~80), DL : 드라이브 번호, DH : 헤드 번호, CH : 실린더나 헤드 번호, CL : 섹터 번호
[Return] AH : 에러 코드

(5) 플로피 디스크를 포매팅한다.

[Entry] AH : 5, DL : 드라이브 번호, DH : 헤드 수, CH : 트랙 수
[Return] AH : 에러 코드

(6), (7) 하드 디스크를 포매팅한다.

[Entry] AH : 6 or 7, DL : 드라이브 번호, DH : 헤드 수, CH : 트랙 번호
[Return] AH : 에러 코드
(6)은 포맷시 불량 트랙을 제거하며, (7)은 지정된 트랙부터 포매팅한다.

(8) 드라이브의 파라미터를 구한다. 하드 디스크에서만 사용한다.

[Entry] AH : 8, DL : 드라이브 번호
[Return] DL : 콘트롤러에 접속되어 있는 드라이브 수, DH : 헤드의 끝 번호, CH : 실린더의 끝 번호, CL : 섹터의 끝 번호

(9) 드라이브의 초기화, 이 루틴은 BIOS의 내부에서만 이용된다.

(0A), (0B) 긴 섹터 읽고 쓰기. 하드 디스크에서만 사용한다.

[Entry] AH : 0A(Read) or 0BH(Write), AL : 작업할 섹터 수, DL : 드라이브 번호, DH : 헤드 번호, CH : 실린더 번호, CL : 섹터 번호, BX : 메모리 버퍼의 선두 번지, ES : 메모리 버퍼의 세그먼트
[Return] AH : 에러 코드

(0C) 디스크의 섹터를 검색한다. 하드 디스크에서만 사용한다.

[Entry] AH : 0C(Read), DL : 드라이브 번호, DH : 헤드 번호, CH : 실린더 번호, CL : 섹터 번호
[Return] AH : 에러 코드

(0D) 디스크의 리세트. 하드 디스크에서만 작업한다. (0)과 비슷하지만 하드 디스크에서만 사용한다.

(0E), (0F) 섹터 버퍼의 읽고 쓰기. 하드 디스크에서만 사용한다.

[Entry] AH : 0E(Read) or 0F(Write), DL : 드라이브 번호, DH : 헤드 번호, CH : 실린더 번호, CL : 섹터 번호, BX : 버퍼의 선두 번지, ES : 버퍼의 세그먼트
[Return] AH : 에러 코드

(10) 드라이브의 준비 여부 확인

이 루틴은 DOS가 내부적으로 사용하는 것이며, 하드 디스크가 준비되어 있는지를 조사한다.

(11) 하드 디스크의 재측정

이 루틴은 BIOS가 드라이브를 제어하기 위해 사용된다.

(12), (13), (14) 하드 디스크의 진단(Diagnostic) 이 루틴은 콘트롤러와 드라이브에 대한 진단을 한다.

[Entry] AH : 12=콘트롤러의 RAM 진단, AH : 13=드라이브의 상태 진단, AH : 14=콘트롤러 내부 테스트
[Return] AH : 에러 코드

- **INT 14H** : RS-232C에 대한 인터럽트
- **INT 15H** : 카세트 입출력
- **INT 16H** : 키보드 인터럽트로 3가지의 기능이 있다.

(0) 키보드에서 키 읽기. 이 루틴은 키 입력이 있을 때까지 기다린다.

[Return] AH : 스캔 코드, AL : 아스키 코드

(1) 키보드에서 키 읽기. 이 루틴은 키 입력을 기다리지 않는다.

[Return] Zero flag : 0=키 입력이 없음, Zero flag : 1=키 입력이 있음, AH : 스캔 코드, AL : 아스키 코드

(2) 키보드 플래그를 조사한다.

[Return] AL : 키보드 플래그(각 비트의 기능은 다음과 같다.)
0 : 우측 Shift가 눌려졌으면 1이 된다.
1 : 좌측 Shift가 눌려졌으면 1이 된다.
2 : Ctrl+Shift가 눌려졌으면 1이 된다.
3 : Alt+Shift가 눌려졌으면 1이 된다.
4 : Scroll Lock이 ON이면 1이 된다.
5 : Num Lock이 ON이면 1이 된다.
6 : Caps Lock이 ON이면 1이 된다.
7 : Insert가 ON이면 1이 된다.

- **INT 17H** : 프린터를위해 할당된 인터럽트로 3가지의 기능이 있다.

(1) 1문자를 프린터에 출력한다.

[Entry] AH : 0, AL : 출력할 문자의 아스키 코드, DX : 프린터 번호(0, 1, 2)
[Return] AX : 1=Printer time out error

(2) 프린터를 초기화한다.

[Entry] AH : 1, DX : 프린터 번호
[Return] AH : 프린터 상태(각 비트에 대응)
0 : Printer time out
1, 2 : 사용하지 않음
3 : I/O Error
4 : Selected
5 : Out of Paper
6 : Acknowledge
7 : Print not Busy

(3) 프린터 상태를 읽어들인다.
[Entry] AH : 2, DX : 프린터 번호
[Return] AH : 프린터 상태 (2)와 동일

- **INT 18H** : ROM BASIC을 호출한다.
- **INT 19H** : 디스크에서 부트스트랩 로더를 읽어 들인다. 디스크에 부트스트랩 로더가 없을 경우에는 INT 18H의 ROM BASIC이 호출된다.
- **INT 1AH** : 시간에 관한 인터럽트

(1) 시간을 읽어 낸다.
[Return] CX : 시간의 상위 워드, DX : 시간의 하위 워드

(2) 시간을 쓴다.
[Entry] AH : 1, CX : 시간의 상위 워드, DX : 시간의 하위 워드

- **INT 1BH** : 키보드 브레이크 어드레스. 〈Ctrl〉+〈Break〉로 프로그램을 중단할 때 중단 어드레스를 갖는다.
- **INT 1CH** : Timer Tick 인터럽트. 이것은 초당 18.2번 호출된다. 이 인터럽트는 INT 8H에 의해 호출되며, 보통은 IRET 명령에 의해 아무 것도 하지 않고 되돌아간다.
- **INT 1DH** : 비디오 파라미터 테이블. 이 인터럽트는 비디오 콘트롤러가 사용하는 테이블 어드레스를 가리킴.
- **INT 1EH** : 디스켓 파라미터. 이 인터럽트는 디스크 콘트롤러가 사용하는 파라미터 테이블 어드레스를 가르킴.
- **INT 1FH** : 그래픽 문자 정의. 고해상도 그래픽 모드에서 128 아스키 문자를 정의할 때 메모리의 번지를 지정함.
- **INT 20H** : 프로그램의 종료
- **INT 21H** : DOS 펑션콜

(0) 프로그램의 종료

(1) 키보드에서 1문자 입력
[Return] AL : 입력한 문자의 아스키 코드

(2) 화면에 1문자 출력
[Entry] AH : 2, DL : 출력할 문자의 아스키 코드

(3) 표준 보조장치에서 1문자 입력. 주로 비동기 통신 어댑터에서 1문자 읽는다.
[Return] AL : 입력된 문자의 아스키 코드

(4) 표준 보조장치 출력
[Entry] AH : 4, DL : 출력할 문자의 아스키 코드

(5) 프린터 출력
[Entry] AH : 5, DL : 출력할 문자의 아스키 코드

(6) 콘솔 I/O
[Entry] AH : 6, DL=FF : 키보드에서 입력(AL), DL〈FF : DL의 내용을 화면에 출력. 입력시 화면에 에코하지 않으며 〈Ctrl〉+〈C〉를 체크하지 않는다.

(7) 에코없는 콘솔 입력
[Return]AL : 입력된 문자의 아스키코드 입력시 화면에 에코하지 않으며 〈Ctrl〉+〈C〉를 체크하지 않는다.

(8) 〈Ctrl〉+〈C〉 체크로 에코없는 콘솔 입력
(7)과 같으나 〈Ctrl〉+〈C〉를 체크함.

(9) 문자열 출력
[Entry] AH : 9, DX : 문자열의 선두번지
문자열의 끝에는 '$'를 붙인다.

(0A) 문자열 입력
[Entry] AH : OA, DX : 버퍼의 선두번지
[Return] DX로 지정된 버퍼에 문자열이 들어간다.

(0B) 키보드 입력상태 확인. 〈Ctrl〉+〈C〉를 체크한다.
[Return] AL=FF : 입력이 있음, AL=0: 입력이 없음.

(0C) 키보드 버퍼를 지우고 키보드 입력루틴 호출
[Entry] AH : 0C, AL : 펑션 번호
[Return] 각 펑션에 준함

이 루틴은 키보드 버퍼를 지운 다음 문자(열) 입력루틴을 호출한다. 예를 들어 키보드 버퍼를 지운 다음 1문자를 입력받고자 할 경우에는 AH에 0CH, AL에는 '1문자 입력' 펑션인 1을 넣고 INT 21H를 호출한다.

(0D) 디스크의 리세트

(0E) 디스크의 지정
[Entry] AH : 0E, DL : 드라이브 번호 (0=A, 1=B)
[Return] DX로 지정된 버퍼에 문자열이 들어 간다.

(0F) 이미 존재하는 화일의 오픈
[Entry] AH : 0A, DX : FBC의 선두번지
[Return] AL : 0=성공, FF=실패

(10) 화일을 닫는다.
[Entry] AH : 10, DX : FBC의 선두번지
[Return] AL : 0=성공, FF=실패

(11) 조건에 맞는 화일찾기
[Entry] AH : 11, DX : 오픈되지 않은 화일의 FBC의 선두 번지
[Return] AL : 0=성공, FF=실패

(12) 조건에 맞는 다음 화일찾기
[Entry] AH : 12, DX : 오픈되지 않은 화일의 FBC의 선두 번지
[Return] AL : 0=성공, FF=실패

(13) 화일의 삭제
[Entry] AH : 11, DX : 오픈되지 않은 화일의 FBC의 선두번지
[Return] AL : 0=성공, FF=실패

(14) 순차적으로 읽기
[Entry] AH : 14, DX : 화일의 FBC의 선두 번지
[Return] AL : 0=성공, 1=End of File(data 없음), 2=DTA Segment too small for record, 3=End of File

(15) 순차적 쓰기
[Entry] AH : 15, DX : 화일의 FBC의 선두 번지
[Return] AL : 0=성공, AL : 1=Disk Full, 2=DTA Segment too small

(16) 화일의 생성
[Entry] AH : 16, DX : 오픈되지 않은 화일의 FBC의 선두번지
[Return] AL : 0=성공, FF=Directory Full

(17) 화일명의 변경
[Entry] AH : 17, DX : 변경할 화일의 FBC의 선두번지
[Return] AL : 0=성공, FF=실패

(18) 이 펑션은 DOS에서 내부적으로 사용한다.

(19) 로딩 디스크의 번호 취득

[Entry] AH : 19

[Return] AL : 디스크 번호(0=A, 1=B)

(1A) DTA(Disk Transfer Address)의 위치 설정

[Entry] AH : 1A, DX : DTA Address

(1B) 디폴트 드라이브에 대한 FAT(File Allocation Table) 정보 취득

[Entry] AH : 1B

[Return] BX : FAT Address, DX : 클러스터 수, AL : 클러스터당 섹터 수, CX : 섹터당 바이트 수

(1C) 지정한 드라이브에 대한 FAT(File Allocation Table) 정보취득

[Entry] AH : 1C, DL : 드라이브 번호

[Return] BX : FAT Address, DX : 클러스터 수, AL : 클러스터당 섹터 수, CX : 섹터당 바이트 수

(1D)~(20) DOS에 내부적으로 사용한다.

(21) 랜덤하게 읽는다.

[Entry] AH : 21H, DX : FCB Address

[Return] AL : 0=성공, 1=End of File(data 없음), 2=DTA Segment too small for record, 3=end of File

(22) 랜덤하게 쓴다.

[Entry] AH : 22H, DX : FCB Address

[Return] AL : 0=성공, 1=Disk full, 2=DTA Segment too small for record

(23) 화일의 크기를 계산한다.

[Entry] AH : 23, DX : 오픈되지 않은 화일의 FBC의 선두 번지

[Return] AL : 0=성공, FF = 화일이 발견되지 않음

(24) 랜덤 레코드 필드 설정

[Entry] AH : 24, DX : 오픈되지 않은 화일의 FBC의 선두 번지

(25) 인터럽트 벡터의 설정

[Entry] AH : 25, AL : 인터럽트 번호, DX : 새로운 처리 번지

(26) 새로운 프로그램 세그먼트의 생성

(27) 랜덤 블록읽기

[Entry] AH : 27, DX : FBC의 선두 번지, CX : 읽을 블록번호

[Return] AL : 0=성공, 1=End of File(data 없음), 2=Not enough space in DTA Segment

(28) 랜덤 블록쓰기

[Entry] AH : 28, DX : FBC의 선두번지, CX : 블록번호

[Return] AL : 0=성공, 1=Disk full, 2=Not enough space in DTA Segment

(29) 화일명의 해석

[Entry] AH : 29, DS.SI : 화일명의 저장 번지, ES.DI : FCB의 이름 저장 영역

(2A) 날짜 읽기

[Entry] AH : 2A

[Return] CX : 년-1980, DH : 월, DL : 일

(2B) 날짜 설정

[Entry] AH : 2B, CX : 년-1980, DH : 월, DL : 일

[Return] AL : 0=성공, FF = 날짜 지정이 잘못 되었음

(2C) 시간 읽기

[Entry] AH : 2C

[Return] CH : 시(0~23), CL : 분, DH : 초, DL : 초 이하

(2D) 시간 설정

[Entry] AH : 2D, CH : 시(0~23), CL : 분, DH : 초, DL : 초 이하

[Return] AL : 0=성공, FF=시간 지정이 잘못 되었음

(2E) Verify 세트 / 리세트

[Entry] AH : 2E, AL : 1=Verify ON, 0=Verify OFF

[Return] AL : 0=성공, FF=날짜 지정이 잘못 되었음

(2F) DTA(디스크 전송 번지)를 구한다.

[Entry] AH : 2F

[Return] ES.BX : DTA Address AL : 0=성공, FF=날짜 지정이 잘못 되었음

(30) DOS 버전을 구한다.

[Entry] AH : 30

[Return] AL : 버전의 정수부분, AH : 버전의 소수부분

(31) 프로그램을 램에 상주하고 실행을 마친다.

[Entry] AH : 31, AL : Parent 프로세서에 전달할 인수, DX : 사용하는 메모리의 크기

(32) 이 펑션은 DOS의 내부에서 사용한다.

(33) 〈Ctrl〉+〈Break〉의 체크 여부의 조사와 설정

[Entry]AH : 33, AL : 0=〈Ctrl〉+〈Break〉의 체크 상태를 조사한다. 1=〈Ctrl〉+〈Break〉를 체크하는 것을 설정한다. DL : 0=체크하지 않는다, 1=체크한다. (AL=1일 때만 리턴된다.)

[Return] DL : 0=OFF, 1=ON(이것은 위의 AL=0일 때만 리턴된다.)

(34) DOS 내부에서 사용한다.

(35) 인터럽트 벡터를 구한다.

[Entry] AH : 35, AL : 인터럽트 번호

[Return] ES.BX : 인터럽트 벡터

(36) 디스크의 스페이스를 조사한다.

[Entry] AH : 36, DL : 드라이브 번호

[Return] AX : FFFF=드라이브의 번호가 잘못 지정됨, AX : 클러스터당 섹터 수, BX : 사용 가능한 클러스터 갯수, CX : 섹터당 바이트수, DX : 클러스터의 총 수

(37) DOS에서 사용한다.

(38) 나라별 정보를 구한다 / 설정한다.

[Entry] AH : 38, DX : 정보를 수록할 메모리 번지, AL : 0=구한다, 1=설정한다.

[Return] DX : 나라별 정보가 세트. 메모리는 32바이트 크기여야 하며 그것의 할당은 다음과 같다.

2바이트 : DATE / TIME format(0=USA, 1=유럽, 2=동양식)

1 : 나라별 코드

1 : 사용치 않음(0이 된다)

1 : ','기호(수치의 3자리 마다 찍는 comma) 아스키 코드

1 : 0

1 : '.'기호(소숫점) 아스키 코드

1 : 0

24 : 내부에서 사용한다.

(39), (3A), (3B) 서브디렉토리의 작성 / 삭제 / 변경

[Entry] AH : 39=작성, 3A=삭제, 3B=변경 DS.DX : 서브디렉토리 이름이 저장된 영역의 번지
[Return] Cy : 0=성공, 1=에러 있음. AH : 3=Path 지정이 잘못 되었다, 5=액세스 실패

(3C) 화일작성
[Entry] AH : 3C, DS.DX : 화일명이 저장된 번지, CX : 화일의 속성
[Return] AL : 3=Path의 지정이 잘못됨, 4=너무 많은 화일이 오픈됨, 5=Disk Full이거나 Read-Only 화일이 같은 이름으로 존재함

(3D) 화일 오픈
[Entry] AH : 3D, DS.DX : 화일명이 저장된 번지, AL : 화일 오픈 목적 AL : 0=읽기 위함, 1=쓰기 위함, 2=읽고 씀
[Return] Cy : 에러 발생 유무((42)와 동일)

(3E) File Handle Close

(3F), (40) 화일이나 다른 장치에서 Read / Write
[Entry] AH : 3F=Read, 40=Write, DS.DX : 데이터 버퍼번지, CX : 읽을 바이트 수, BX : 화일 핸들
[Return] Cy : 에러 여부, AL : 5=액세스 실패, 6=핸들이 부적당함, AX : 읽어들인 바이트 수(Cy=0일 때)

(41) 화일 삭제
[Entry] AH : 41, DS.DX : 화일명이 저장된 번지
[Return] Cy : 에러 여부, AL : 2=File Not Found, 5=액세스 실패

(42) 화일 포인터의 이동

(43) 화일속성 조사 / 변경
[Entry] AH : 43, AL : 0=속성조사, 1=변경, DS.DX : 화일명이 저장된 번지, CX : 새로운 속성(AL=1일때)
[Return] Cy : 에러 여부, AL : 2=File Not Found, 3=Path Not Found, 5=액세스 실패, CX : 화일의 속성 (Entry의 AL=1일 때)

(44) I / O 콘트롤

(45), (46) File Handle의 2중화

(47) 지정한 드라이브의 커런트 디렉토리 구하기
[Entry] AH : 47, DS.SI : 64바이트 버퍼번지, DL : 드라이브 번호
[Entry] Cy : 에러 여부, AL : 15=드라이브의 잘못 지정, DS.SI : 커런트 디렉토리 문자열

(48), (49) 메모리 할당 / 해제

(4A) 메모리 블록의 축소 / 확대

(4B) 프로그램의 로드 혹은 로드후 실행

(4C) 프로그램의 중단

(4D) 서브 프로세서에서의 인수를 취득

(4E) 조건에 맞는 화일을 찾는다.
[Entry] AH : 4E, DS.DX : 화일명이 저장된 번지, CX : 화일의 속성
[Return] Cy : 1⇒AL : 2=발견되지 않음, 더 이상 화일이 없음, Cy : 0 ⇒DTA에 저장된다.(22바이트 크기)
1바이트 : 화일 속성
2 : 화일이 작성된 시간
2 : 화일이 작성된 날짜
2 : 화일 크기(하위 워드)
2 : 화일 크기(상위 워드)
13 : 화일명과 확장자

(4F) 조건에 맞는 다음 화일을 찾는다. ((4E)와 동일)

(50)~(53) DOS에서 사용

(54) Verify 상태를 조사한다.
[Return] AL : 0=OFF, 1=ON 되어 있음

(55) DOS에서 사용한다.

(56) 화일명의 변경
[Entry] AH : 56, DS.DX : 새로운 화일 이름이 저장된 번지, ES.DI : 현재의 화일명
[Return] Cy : 에러 여부, AL : 3=Path Not Found, AL : 5=액세스 실패, 17=다른 드라이브로 재명명하려 했음

(57) 화일이 작성된 날짜, 시간의 취득 / 변경

(58) DOS에서 사용

(59) 확장된 에러를 구한다.

(60) 임시 화일의 생성

(5B) 새로운 화일의 생성. 펑션 3C와 동일 하나 지정한 화일이 이미 존재할 경우에는 화일을 만들지 않는다.

(5C) 화일의 LOCK과 UNLOCK

- **INT 22H** : DOS에서 사용한다.
- **INT 23H** : 〈Ctrl〉+〈Break〉로 프로그램을 중단시켰을 때 제어가 전송되는 어드레스를 구한다.
- **INT 24H** : 에러코드를 구한다.

[Return] AH : 0=디스크 에러, 1=기타 에러, DI : 에러 코드
0 : 프로텍트 탭(Protect Tab)이 붙어 있다.
1 : 알 수 없는 단위이다.
2 : 디스크 준비상태가 잘못되었다.
3 : 명령어가 잘못되었다.
4 : 패리티 체크가 잘못되었다.
5 : 요구 구조 길이가 다르다.
6 : 탐색 에러
7 : 미디어의 종류를 알 수 없다.
8 : Sector Not Found
9 : Printer Paper Out
A : 기록 에러
B : 쓰기 에러
C : 디스크가 포맷되지 않았다.

- **INT 25H** : 절대 디스크 읽기

[Entry] AL : 드라이브 번호, CX : 읽을 섹터 수, DX : 첫번째 논리섹터 번호, DS.BX : 메모리 버퍼의 선두 번지
[Return] Cy : 에러 여부(AH : 에러 코드)

- **INT 26H** : 절대 디스크 쓰기. INT 25와 동일하다.
- **INT 27H** : 프로그램을 메모리에 상주하고 종료시킴
- **INT 28H~2EH** : DOS 내부에서 사용
- **INT 2FH** : 두 프로세서간의 통신
- **INT 30H~FFH** : 차후를 위해 보유되거나 베이직 등에서 사용한다.

이들 중 60H~67H는 사용자를 위해 예약되며, 33H는 마우스가 있는 시스템에서는 마우스를 위해 지원된다. 마우스에 관한 사항은 본지 5월호를 참고하기 바란다.

프로그램의 작성과 실행

박연진 / 여주컴퓨터학원 강사

지난호에서는 터보 베이직이 무엇이며 기존 베이직 언어와는 어떤 차이점을 갖고 있는지에 대해 살펴 보았다. 이번호부터는 터보베이직을 배워 나가는데 있어 가장 기초가 되는 기본적인 명령문부터 이 명령문들이 어떤 형태로 사용되는지 살펴보기로 하자.

PRINT문의 기능

PRINT문은 계산 결과등을 화면에 출력하는 의미로 다음과 같은 기능이 있다.

① 연산결과를 화면에 출력한다.
② 문자열을 화면에 표시한다.
③ 한 줄을 띄운다.(PRINT문 다음에 아무것도 기술하지 않으면 1줄을 띄운다.)
④ 변수의 값을 화면에 출력한다.

베이직의 연산자

연산이란 연속된 계산이라는 의미로 컴퓨터에서는 '계산'이라는 말 대신'연산'이라는 표현을 하며, '+', '−' 등의 부호를 연산자라고 한다. 베이직의 연산자는 산술연산자와 논리연산자, 관계연산자가 있는데 산술연산자는 '+, −, *, /, ^, \, MOD'등 7가지가 있다. (다른 연산자는 다음에 배우게 된다.)

+ : 더하기
− : 빼기
* : 곱하기
/ : 나누기
^ : 거듭제곱
\ : 정수의 나눗셈(이 기호는 컴퓨터에 따라 화폐기호인 '₩'로 대치되기도 한다. 정수 나눗셈의 몫은 소수 이하가 잘려져 정수로 된다.)

(예) 7/4 ---> 1.75 ---> 1

MOD : 정수의 나눗셈에서 나머지를 구한다.

(예) 10 MOD 3 ---> 1

이들 연산자가 하나의 식에서 사용되는 경우 우선순위는 다음과 같다.

()→ ^ → −(마이너스 부호)→ *, / →\, MOD→+, −

문자열

문자열은 스트링(String)이라고도 하며, 겹따옴표(" ")로 묶은 부분을 말한다. 따옴표 안에는 다른 따옴표가 포함될 수 없으며, 키보드에서 입력할 수 있는 모든 문자를 포함할 수 있다. 다음은 문자열의 예이다.

```
"KOREA"
"He said,'I love you.'."
"안녕하십니까?"
```

변수

변수(Variable)란 어떤 값을 기억하는 기억장소를 말한다. 변수는 수치자료를 기억하는 수치변수와 문자자료(문자열)를 기억하는 문자변수로 구별되며, 변수이름은 사용자가 자유로이 지정할 수 있으나 여기에는 몇가지 규칙이 있다.

① 변수명의 길이에는 제한이 없고, 첫문자는 반드시 A~Z의 영문자(소문자라도 상관없다)로 시작해야 한다.
② 2번째 이후의 문자들은 영문자나 0~9의 수치와 '.'로 이루어져야 한다.
③ 문자변수를 나타낼 때에는 변수명의 뒤에 $를 붙인다.
④ 예약어 (베이직의 명령어)를 포함할 수는 있지만 예약어 자체를 변수명으로 사용해서는 안된다. 특히 다음의 예약어를 변수명으로 사용하지 않도록 주의하라.

ON, AS, IF, TO, OR, FOR, GET, PUT

다음은 올바르게 사용한 변수명의 예이다.

KOREA, A, A123$, B.19, ABC, K5, C, A.12.BC$, ABC$

다음은 잘못 사용된 변수의 예이다. ()안의 숫자는 위의 조건이다.

A-19(2), A$B(3), AS(4), ON$(4), 9AB(1), $123(1)

대입문(치환문)

베이직에 있어 등호(=)의 의미는 수학에서의 '같다'의 의미가 아니라 대입을 나타낸다.

A=3*4

위의 식의 해석은 A와 3*4가 같다

는 의미가 아니라 3*4의 연산 결과를 A라는 변수에 저장하라는 의미이다. 그러므로 등호의 좌변에는 반드시 단 하나의 변수가 나타나야 하며, A+5=10이나 3*4=A이라는 표현은 잘못 된 것이다. 위의 A=3*4라는 식은 A에 12라는 값이 저장될 것이므로 결국 A와 3*4가 같다고 볼 수 있지 않느냐고 반문할 지 모르지만 다음의 식을 보자.

```
A=A+1
```

수학에서는 위의 식이 성립하지 않는다. 그러나 컴퓨터에서는 A에 1을 더하여 그 결과를 다시 A에 기억시키라는 의미가 된다. 따라서 A의 값이 증가하게 된다. 결과적으로 이것은 변수 A의 값을 1만큼 증가시키라는 의미가 된다. 그러면 다음의 식들이 어떤 의미를 갖는지 각자 생각해 보기로 하자.

```
A=A-B
A=A*3-4
A=A/10+5
A=A\4+5
```

이와같이 연산결과를 변수 자신에게 되돌리는 것을 '누적시킨다'고 말한다.

이제 기초적인 지식은 준비된 상태이므로 본격적인 프로그래밍 훈련으로 들어가 보자. 여기에 지름길이란 없으며 많이 작성하고, 많이 모방하고, 많이 개조해 보는 것 이외의 방법은 없다. 그러므로 프로그래머의 실력은 그 사람이 프로그램 훈련에 쏟는 시간에 비례한다.

앞으로는 예제를 가지고 프로그램 작성요령을 익히게 되며, 이것은 일련번호를 붙일 것이므로 독자들도 〈P-1.BAS〉, 〈P-2.BAS〉,…와 같이 일련번호를 붙여 프로그램을 작성해 두면 나중에 좋은 자료가 될 것이다.

〈예제 1〉

밑변이 21㎝, 높이가 19㎝인 삼각형의 넓이를 구하라.
넓이=밑변*높이/2

〈처리 방법〉

컴퓨터에 어떤 자료를 기억시켜 둘 때는 반드시 변수를 이용하여야 한다. '21을 기억하라'라는 막연한 표현은 컴퓨터가 처리하지 못하게 된다. '어떤 변수에 21을 기억시켜라'와 같이 명확하게 표현하여야 한다.

```
<P-1.BAS>
1 '******************'
2 '  삼각형의 넓이   '
3 '******************'
4 A=21
5 B=19
6 C=A*B/2
7 PRINT C
8 END
[실행결과]
 199.5
```

〈프로그램 설명〉

1~3 : 주석문
4 : 밑변을 변수 A에 기억시킨다.
5 : 높이를 변수 B에 기억시킨다.
6 : A의 값과 B의 값을 상대로 연산을 하여 그 결과를 변수 C에 기억시킨다.
7 : 변수 C의 값을 출력한다. (이때 PRINTC 처럼 변수를 PRINT문에 붙여쓰면 안된다.)
8 : 프로그램의 실행을 마친다.

PRINT문과 함께 쓰이는 부호

위의 프로그램은 실행결과 199.5를 출력한다. 이 값은 다른 사람이 봤을 때 무엇을 나타내는 수치인지 전혀 알 수 없다.

삼각형의 넓이=199.5㎠

위와 같이 출력이 된다면 더욱 좋을 것이다. PRINT문은 1줄에 여러개의 자료를 출력할 수 있는 기능이 있다. 이때는 자료와 자료 사이를 콤마(,)나 세미콜론(;)으로 구분해 주면 된다. 이들 부호는 다음과 같은 기능을 한다.

, : 다음의 자료를 일정한 간격(8문자 이내)을 유지하여 출력한다.
; : 다음의 자료를 붙여서 출력한다.

위에서 (,)는 다음의 탭 존으로 이동하는 기능을 갖는데 베이직의 경우에는 매 8컬럼마다 탭존이 설정되어 있으므로 현재 출력된 위치에서 우측의 가장 가까운 탭존으로 이동하게 된다. 위의 7행을 다음과 같이 바꿔보자.

```
7 PRINT "삼각형의 넓이 = ";C;"cm² "
```

위의 명령은 '삼각형의 넓이='를 출력한 다음 붙여서 변수 C의 값을 출력하고, 또 붙여서 '㎠'을 출력하라는 의미이다.('²'는 〈Alt〉키를 누른상태에서 2, 5, 3을 차례로 입력하고, 〈Alt〉키에서 손을 떼면 출력된다.) 이번에는 문자 자료의 경우를 알아보자.

〈예제 2〉

문자변수에 SEOUL, KOREA, GOOD, MORNING을 기억시키고, 다음과 같이 출력하는 프로그램을 작성하라.

```
[실행결과]
SEOUL                   KOREA
KOREA                   SEOUL
SEOULKOREA
KOREASEOUL
GOOD MORNING.
MORNING GOOD.
```

```
<P-2.BAS>
1 REM ================
2 REM 문자 변수의 활용
3 REM ================
4 A$="SEOUL"
```

```
5 B$="KOREA"
6 C$="GOOD"
7 D$="MORNING"
8 ? A$,B$
9 ? B$,A$
10 ?A$;
11 ?B$
12 ?B$;A$
13 ?C$;" ";D$;"."
14 ?D$;" ";C$;"."
15 END
```

〈프로그램의 설명〉

1~3 : 주석문, 작은 따옴표(')는 REM 문과 의미가 같으며, REM은 Remark를 나타내는 베이직의 명령어이다. 요즈음에는 REM보다 (')를 사용한다.

4~7 : 문자 변수에 값을 할당한다. 문자열은 반드시 따옴표(" ")로 묶는다.

8~12 : 변수의 내용을 출력한다. PRINT 문은 약어로 ?로 대치될 수 있다.

10~11 : PRINT문이 (;)이나 (,)로 끝나면 줄을 바꾸지 않으므로 다음에 출력될 내용이 그 줄에 출력된다. 그러므로 10행과 11행을 9행처럼 붙여서 기술해도 되고 따로 기술해도 된다.

12~15 : 변수의 사이에 공백을 출력하는 보기이다. 공백도 따옴표(" ")로 묶으면 문자로 취급된다.

〈예제 3〉

반지름 15㎝인 원의 둘레와 넓이를 구하라.
둘레 : $2\pi r$
넓이 : πr^2

〈처리 방법〉

컴퓨터에는 π라는 문자가 있지만 이 문자는 3.14159...를 의미하지는 않는다. 그러므로 우리는 π대신에 변수를 설정해서 처리해야 한다.

〈P-3.BAS〉

```
'=================='
' 원의 둘레와 넓이 '
'=================='
PI=3.14159
R=15
LENGTH=2*PI*R
AREA=PI*R^2
PRINT "반지름 =";R
PRINT "둘레  =";LENGTH,"넓이 =";AREA
END
```

〈실행 결과〉

```
반지름 =15
둘레 =94.24771    넓이 =706.8578
```

대문자와 소문자

터보베이직은 대문자와 소문자를 구분하지 않으므로 소문자나 대문자 혹은 서로 섞어서 사용할 수도 있다. 따라서 다음의 표현은 모두 같은 것으로 취급한다. (명령어 자체를 띄어 쓰면 안된다.)

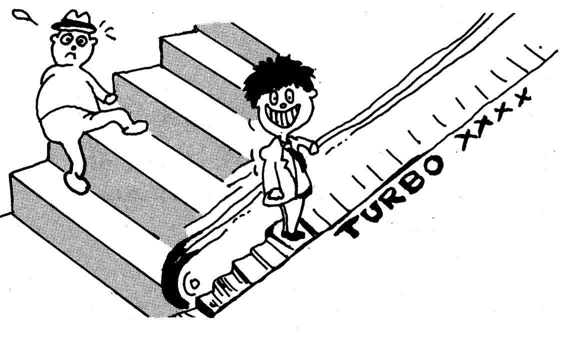
Print, PRINT, pRint, PrInT

대문자를 적절히 섞어서 사용하면 좋지만 그때마다 〈shift〉키를 눌러야 하는 불편함과, 소문자 'l'과 숫자 '1'과의 구별이 모호해지는 등의 문제가 발생할 소지가 있어 모두 대문자로 기술하는 것을 원칙으로 한다. 단 베이직은 다음의 경우에는 대문자와 소문자를 구별하므로 대, 소문자를 구별하여 기술하여야 한다.

① 따옴표(" ")로 묶인 문자열
② 주석문(작은 따옴표 (')나 REM이후의 문장)
③ DATE문 다음의 문자열
④ 키보드에서 입력받은 문자열

베이직을 배운 경험이 있는 독자라면 위의 예제 프로그램을 보고 GW-BASIC 과 다른 부분이 없지 않느냐? 고 반문을 할 것이다. 물론 현재는 그렇다.

터보베이직은 GW - BASIC을 기초로 만든 것이기 때문에 GW-BASIC의 명령어를 90%이상 포함하고 있으며, 10%정도의 명령어는 삭제되었거나 퇴화되었다. 퇴화된 명령어 중에는 LIST, AUTO, RENUM, TRON, TROFF, DELETE, SAVE, LOAD 등이 있다. LIST명령만 보더라도 터보베이직은 항상 에디터 모드로 들어갈 수 있으며, 프로그램을 상하 좌우로 스크롤 시켜가면서 볼 수 있기 때문에 굳이 이런 명령이 존재할 필요가 없는 것이다. 또한 터보베이직은 행번호가 없기 때문에 행번호에 관계되는 AUTO, RENUM, DELETE가 없는 것도 당연한 것이다. 터보베이직의 진가는 횟수가 지남에 따라 차츰 알게 될 것이므로 너무 조급하게 생각하지 말기 바란다.



새로운 교육용 프로그램
「지원 중학(국어, 영어)」 선보여

지원엔지니어링에서는 중학교용 교육프로그램인 지원 중학(국어, 영어)을 선보였다. 이번에 선보인 프로그램은 교과과정과 동일한 순서로 편집되었으며 참고서나 문제지의 기능을 갖춘 매우 유용한 프로그램이라 말할 수 있다.

지난 6월 교육용 PC가 16비트로 확정됨에 따라서 PC의 보급확산과 더불어 큰 문제점으로 대두된 것이 교육용 프로그램의 부재였다.

이러한 때에 발맞추어 지원엔지니어링에서 중학교용 학습프로그램을 선보였다. 이 프로그램은 중학 1, 2, 3 학년용 영어, 수학 과목으로 각 과목당 플로피디스크 3장으로 구성된다.

예전에 발표된 교육용 프로그램들의 내용이 극히 피상적이었으나 이번에 개발된 프로그램은 교과과정과 동일한 순서로 편집되어 참고서나 문제지의 기능을 갖춘 매우 구체적인 프로그램이다.

이 프로그램은 터보베이직, 퀵베이직, 터보 C, 어셈블리 등의 언어로 작성되어 있으며, 애초에 완성형 세운한글카드 상에서 작성된 것을 현재 KS 한글로 개정작업을 행하고 있다.

지원엔지니어링의 프로그램 보급방안으로는 현재 교육용 프로그램이 전무한 상태에서 16비트 PC의 보급확산과 관련하여 저렴한 가격으로 공급할 방침이며, PC를 소지하고 있지 않은 학생들도 학교를 통하여 사용할 수 있게끔 할 계획이다.

프로그램을 제작하는 과정중 가장 중요한 내용선정과정을 살펴보면 일선학교의 전산담당 요원과 선생님들과의 워크숍을 통하여 담당교사들이 문제를 선정하고 지원엔지니어링에서는 프로시듀어를 담당하는 방식으로 하여 내용을 선정하였다.

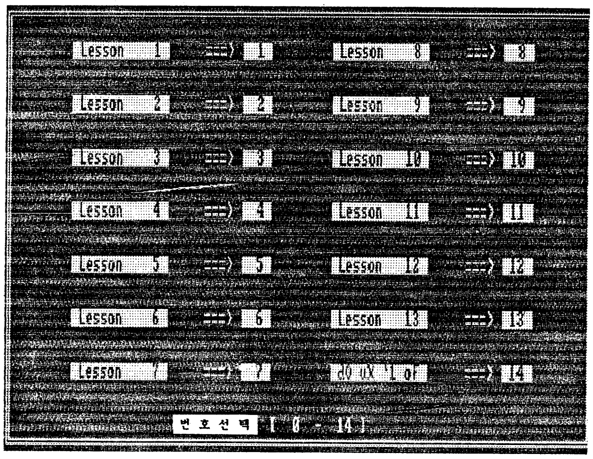
내용의 구성방법은 영어과목인 경우 교과과정에 맞추어 중학교 1학년과정이면 그 과정에 맞는 필수적인 단어공부난을 소개하여 기초적 또는 심화학습을 할 수 있도록 유도하고 있다.

교육용 프로그램은 그 성격상 화면에 그래픽이 많이 출력되며, 이해를 돕기위하여 화면에 대한 설명을 한글로 표시해야 한다. 그리고 지리한 분위기에서 벗어나려면 단조롭지 않게 화면을 구성하여야 한다.

이러한 면에서 볼때 이 프로그램들은 적절한 조건을 구비하고 있다고 볼 수 있다.

우선 처음 프로그램을 본 디스켓으로 부팅시키면 '중학OO''라는 로고와 함께 경쾌한 음악을 들려 준다. 영어과목을 예로 들면 음악이 끝남과 동시에 각 단원을 선택할 수 있도록 화면 1과 같은 메뉴화면이 나온다. 이 메인메뉴에서 각 단원을 선택할 때는 사용자가 선택하여 사용하면 된다. 여기에서 단원을 선택하면 해당 단원으로 넘어가며 각 단원에 해당하는 단어, 숙어 및 문법 등의 요약설명이 커서를 따라 나타난다.

〔화면 1〕 영어 프로그램의 메뉴화면



그리고 단어나 숙어에 해당하는 문제가 객관식, 주관식의 형태로 수록되어 있고 각 단원의 본문을 예문으로 한 문제들도 참고서나 문제지처럼 짜임새 있게 구성되어 있다. 만일 맞는 답을 입력하면 화면 위쪽에 맞음 부분에 점수가 기록되며 틀릴 경우에는 틀림난에 기록된다. 그리고 한 단원이 끝날때마다 문제 갯수당 맞은 갯수와 틀린 갯수, 적중률 등이 표시된다.

국어의 경우에는 목차가 각 문장의 장르별로 수록되어 있다. 즉 시, 수필, 논설문 등으로 그룹지어져 메뉴 화면이 구성되는데 여기서 목차를 선택하면 이와 함께 각 장르의 특징과 구성 등의 요약 설명이 나타난다. 그리고 각 장르에 따른 문제가 역시 주관식, 객관식으로 골고루 수록되어져 있다. 그리고 동사는 현재 개발된 프로그램 이외에 앞으로의 프로그램 개발계획으로 국민학교 4, 5, 6학년용 국어, 산수, 자연과목용 프로그램을 개발할 예정으로 있다.

이렇게 실질적인 교육용 프로그램이 발표된 것은 시기적으로 상당히 유효 적절하다고 할 수 있는데 판매 가격은 이만원으로 책정되어 있다. 자세한 문의사항은 삼우상사(02)691-5677로 하면 된다.

펑션콜을 이용한 화일의 입출력

이용일 / 한국전자통신 연구소

극히 일부의 예외를 제외하면 프로그램은 반드시 어떤 형태로든 입출력을 동반한다. 필터 명령어 등은 차라리 입출력이 명령어처리 수행의 중심이라고 해도 좋을 정도이며, 프로그램을 작성하는 경우에는 디바이스 및 화일과의 입출력에 대한 지식을 빠뜨려서는 안된다. MS-DOS의 XENIX 호환펑션콜을 중심으로 펑션 콜을 사용하여 어떻게 여러가지 형태의 입출력을 실현하는가에 대해서 알아보자.

다른 디바이스의 동격화

MS-DOS의 XENIX 호환 펑션 콜에서의 화일이란, 프로그램이 수행하는 입출력의 대상 모두를 의미한다. 예를 들면, 디스크상에서 데이터를 읽어내는 경우나 프린터로 데이터를 출력하는 것도 모두 화일로의 액세스라고 생각할 수 있는 것이다. 이와같이 모든 입출력의 대상을 동격으로 취급하는데서 얻는 장점은 명료하다. 프로그램측에서는 모든 입출력을 통일적으로 취급할 수 있고, 사용자는 필요에 따라 입출력의 지정을 임의의 화일 및 디바이스로 새로이 바꾼다든지 (I/O 재지정), 또는 어떤 명령어의 출력을 그대로 다른 명령어의 입력으로 한다든지 (파이프)하는 것처럼 명령어를 활용할 수 있는 점 등이 그 장점인 것이다.

입출력의 대상을 화일이라고 하는 통일적인 개념으로 처리할 수 있는 것은 사용자와 프로그램작성자 모두에게 이익을 가져다 주지만, 물리적으로 실체가 다른 것을 무리하게 동격화하고 있기 때문에 양쪽 모두를 완전히 동등하게 취급하는 것은 아니다. 세밀한 처리를 하는 경우에는, 디바이스마다 다른 대응책이 필요하게 된다. 따라서 프로그램 작성자에게는 동격화가 어떤 방법으로 수행되는가 하는 문제와 개개의 차이는 어떤 부분에서 나오는가 하는 의문인 "화일이란 무엇인가"라는 지식이 요구된다. 그래서 디스크상의 화일과 동등하게 취급되는 디바이스를 "디바이스 화일", 또 디스크상의 화일을 의미하는 경우에는 "디스크 화일"이라 부르며 디바이스 화일과 디스크 화일을 합쳐서 "화일"이라고 구분하기로 하자.

(1) 디스크 화일과 디바이스 화일

디바이스 화일과 디스크 화일의 동격화의 개념을 이해하기 위해서는 먼저 디바이스 화일과 디스크 화일이 물리적(하드웨어적)으로 어떻게 다른가를 이해할 필요가 있다. 디스크 장치의 큰 특징으로서는 디스크상의 임의의 데이터를 액세스할 수 있다는 것을 들 수 있다. 예를 들면, 한 번 읽어낸 데이터는 그 데이터를 소거한다든지, 고쳐 쓴다든지 하지 않으면 몇번이라도 임의의 부분에서 다시 읽어낼 수 있고, 써넣은 데이터에 대해서도 나중에 부분적으로 수정할 수 있다. 즉 디스크상의 데이터는 직접(Random) 액세스가 가능한 것이 큰 특징이다. 그와같은 자유도가 있는 만큼, 디스크 액세스를 할 때에는 관리해야할 정보가 많은 것도 특징이라고 말할 수 있을 것이다.

한편 디바이스 화일인 경우에는, 모든 데이터는 선두부터 차례로 한 바이트 단위로 입출력한다. 임의의 부분을 자유로이 읽어낼 수 없으면, 일단 출력한 데이터를 다시 쓰는 것도 불가능하다. 이와같이 디바이스 화일의 경우에는 순차(Sequential)접근 이외의 방법으로는 액세스가 불가능(자유도가 낮다)하지만, 입출력할 때에 관리하는 정보는 디스크 화일과 비교하면 대단히 작게 되어 있다.

(2) 디스크 화일과 디바이스 화일의 동격화

실제의 프로그램이 수행하는 화일의 액세스는 대개 3가지로 구별할 수 있다.

- 1 패스 순차 액세스
- 복수 패스 순차 액세스
- 직접(Random) 액세스

첫번째의 것은 데이터의 선두부터 차례로 읽고 쓰는 경우에 한정되는 경우이다. 필터 명령어등 대부분의 명령어가 여기에 해당한다. 디바이스 화일의 경우는 이 방법으로 밖에는 액세스할 수 없다. 두번째의 경우는 첫번째의 것과 대체로 같지만 —즉, 선두부터 차례로 읽어 들인다— 필요에 따라 읽어내는 것을 여러번 할 수 있는 것이다. 예를 들면, MS-DOS의 MASM은 화일을 순차적으로 읽어 들이지만, 오브젝트 코드를 생성하기까지는 화일을 두번 읽어 들이게 된다. 세번째의 경우는 화일의 임의의 위치부터 임의의 길

이의 데이터를 액세스하는 경우이다. 데이터베이스의 데이터 등이 이 유형의 액세스 방법이다.

이들 중 두번째와 세번째는 디바이스 화일에 대해서만 할 수 있는 액세스 방법이지만, 실제의 화일 액세스방법 중에서 차지하는 비율은 작으며, 대부분의 액세스는 첫번째의 방법으로 수행된다. 예를 들면, 에디터의 텍스트를 읽고 쓰는 것 및 COPY 명령어의 화일 전송, 또는 DOS에 의한 실행 프로그램의 읽어들이기 등은 모두 이 1패스 순차 액세스 방법을 이용하고 있다. 그러므로, 화일 액세스의 펑션 콜에서도 디바이스 화일과 같은 형식으로 순차 액세스를 하도록 해 놓으면, 첫번째의 액세스방법을 취하는 한, 디바이스 화일과 디스크 화일의 동격화가 가능하게 된다.

다음에 이 방법을 확장하여 두번째 및 세번째에 해당하는 액세스 방법을 지원하는 것도 생각할 필요가 있다. 이들이 첫번째와 다른 것은 임의의 위치에서부터 액세스하는지 어떤지 하는 것이다. 여기서 두번째의 방법은 일단 화일의 끝까지 읽어들이면, 다시 화일의 선두 혹은 임의의 위치부터 다시 읽어들인다고 볼 수 있다. MS-DOS에서는 디바이스 화일의 액세스를 위해서 액세스위치(화일의 선두 및 최후, 현재의 액세스 위치부터의 옵셋)를 지정하기 위한 화일을 준비하여 이 문제를 해결한다. 이상을 정리하면, 다음과 같이 요약할 수 있다.

- MS-DOS에서는 디스크 화일과 디바이스 화일은 모두 "바이트 데이터의 흐름"으로 취급된다.
- MS-DOS의 기본적인 화일 액세스는 순차 액세스이다. 대부분의 명령어에서는 이 제한이 모순을 야기시키지 않는다.
- 디스크 화일 특유의 직접 액세스를 지원하기 위해서 액세스위치를 지정하는 화일 포인터의 이동을 위한 펑션 콜을 준비하고 있다. 그러나 이 펑션 콜은 디바이스 화일에 대해서는 이용할 수 없다.

화일핸들에 의한 액세스

화일의 액세스에는 데이터 이외의 여러가지 정보 관리를 동반한다. 예를 들면 화일의 실체가 디바이스인지 또는 디스크 화일인지, 어떤 드라이브에 저장되어 있는지, 다음의 액세스 위치는 어디인지 등의 정보를 관리하고 있지 않으면 화일을 액세스할 수 없다.

CP/M 및 MS-DOS의 CP/M 호환에서는 그들 관리정보(FCB라 불리는 수십바이트의 구조체)는 프로그램측에서 관리하고 있다. 그러나, FCB를 프로그램측에서 관리하는 것은 프로그램을 작성할 때의 부담을 쓸데없이 증대시킨다. 또, FCB는 계층화 디렉토리에 대응한 구조가 아니기 때문에 버전 2.0 이후의 MS-DOS에서는 DOS가 제공하는 기능에 대응할 수 없다.

이들 문제를 해소하기 위해, 버전2.0 이후의 MS-DOS에서는 XENIX호환콜이 지원되도록 되어 있다. 이것은 UNIX에 준비되어 있는 화일 액세스용의 펑션 콜과 거의 같은 것이며, FCB를 이용하는 경우에 발생하는 문제점은 자연히 해소되게 된다.

펑션 콜로 개방된 화일에는 MS-DOS의 시스템 내부에 관리정보를 기억하는 영역이 준비되고, 프로그램에는 그 화일을 액세스하기 위한 번호가 넘겨진다. 이 번호를 "화일핸들"이라 부른다. 화일핸들은 "화일 액세스를 위한 열쇠"라고 생각할 수 있다. 화일 액세스를 비롯하여, 화일포인터의 이동 및 여러가지 제어등 각 화일에 대하여 수행하는 펑션 콜은 기본적으로 이 화일핸들을 함께 넘겨, 처리의 대상이 어느 화일인가를 지정한다.

화일핸들을 한 워드(2 바이트)의 정수이기 때문에 최대로 65536개의 화일을 동시에 취급할 수 있지만, MS-DOS의 경우에는 한 프로세스로 개방(open)할 수 있는 최대의 화일갯수를 20으로 제한하고 있다. 또, 모든 프로세스가 개방하고 있는 화일 수의 합계도 MS-DOS를 기동할 때의 CONFIG.SYS 화일의 FILES 명령어 지정에 의해 제한된다. 디폴트는 버전2.0에서는 5(버전3.1에서는 8)이지만, 지정에 의해 버전2.0에서는 최대 99, 버전3.1에서는 최대 255까지 지정할 수 있다.

이와같은 제한은 MS-DOS에 있어서 화일핸들의 이행문제에 기인하기 때문이다. 앞에서 설명한 대로 화일핸들을 이용하는 경우, MS-DOS측에서 화일 액세스에 관한 관리정보를 기억하게 된다. 이 관리정보(FCB와 유사한 구조체, 이후 시스템용 FCB)의 기억영역은 MS-DOS 기동시에 그 크기가 결정되고, 이후에는 변환하지 않기 때문에 한번에 개방할 수 있는 화일수에도 제한이 가해지기 때문이다.

화일핸들의 값은 0~19의 어느 하나이고, 그 값은 MS-DOS에 의해 결정되어 프로세스로 돌려지지만, 그 중에서 화일 핸들 0~4를 "표준 화일핸들"이라 부르고, 각각의 값에 특별한 화일핸들을 지정하고 있다. 이들 화일핸들은 프로그램이 기동한 시점에서 이미 개방되어 있기 때문에 간단한 필터 명령어와 같은 프로그램을 작성하는 경우에는 화일 개방 등의 처리를 생략한 프로그램을 작성하는 것도 가능하다. 특히 화일핸들의 0과 1은 명령어라인에서 재지정을 지정하면 자동적으로 화일이 개방되고, 그 화일핸들이 할당되기 때문에 그 기능을 이용하기 위해서도, 처리하는 데이터의 입출력은 원칙적으로 표준 화일핸들을 이용하게 된다. 또 표준 에러출력은 디폴트로 CON 디바이스(화면)에 대응하고, 사용자에 대하여 에러메시지를 출력하는 데 이용된다.

화일 핸들로부터 화일 실체까지의 경로

화일 핸들이란 각 화일에 고유한 번호는 아니다. 즉, 1:1 대응이 아닌 것이다. "화일핸들이 n이기 때문에 어떤

화일"이라는 사고방식은 엄밀히 말하면 정확한 표현이 아니다. 화일핸들과 화일 실체의 대응관계는 조금 복잡하지만 이를 이해하면 화일핸들에 관한 거동을 이해할 수 있으므로 복잡한 이용방법도 잘 활용할 수 있게 된다.

펑션 콜에서 이용하는 화일핸들은 MS-DOS의 내부에서 일단 시스템용 FCB로 대응시키고, 그 시스템용 FCB의 정보에 따라서 실제의 화일이 액세스된다. 따라서 화일핸들은 그림 1에 보인 것처럼 내부에서 2단계의 변환을 받게 된다.

화일핸들과 시스템용 FCB의 대응은 PSP에 기억된 변환표에 의해 수행된다. 여기서 변환표는 PSP의 시스템 사용영역 일부에 위치하고 있다. 즉, 화일핸들은 각 프로세스마다 고유한 것이고, 프로세스가 다르면 같은 화일핸들이라도 다른 화일을 지정할 가능성이 있다고 하는 것이다.

또 화일핸들과 시스템용 FCB는 1:1 대응이 아니라 복수:1 대응인 관계이기 때문에 화일핸들이 다르더라도 같은 FCB(같은 화일)를 의미하는 경우가 있다. 예를 들면, I/O 재지정을 이용하지 않는 경우에는 표준출력과 표준 에러출력은 같은 시스템용 FCB(CON 디바이스에 대응한다)에 대응하고 있기 때문에 어느 화일핸들로 출력해도 데이터는 화면에 출력된다.

한편, 시스템용 FCB로부터 물리적인 화일/디바이스로의 대응은 MS-DOS 본체에 의해 FCB 내부의 데이터에 따라서 수행된다. FCB의 내부에는 어느 디바이스 드라이버를 경유하여 화일을 액세스하고 있는지에 대한 정보가 저장되어 있기 때문에 펑션 콜의 내용을 해석하여 디바이스 드라이버가 요구하는 형식으로 고친 후에 디바이스 드라이버를 호출한다. 그 FCB가 대응하고 있는 화일이 디스크상의 것이면, 다음에 액세스하는 위치 즉, 화일 포인터와 디스크의 시스템영역 정보로부터 디스크의 어느 매개체(Media)의 어느 섹터를 액세스하는지를 결정하고, 디스크 화일의 경우에 위치 지정은 불가능 또는 무의미하므로, 펑션 콜에서 지정된 수를 그대로 읽고 쓴다.

화일핸들의 개방/폐쇄

화일핸들은 MS-DOS가 각 프로세스에 할당하는 것이고, 그 할당(화일핸들의 개방)은 펑션 콜에 의해 수행된다. 또, 화일의 액세스가 종료한 후에는 역시 펑션 콜에 의해 불필요하게된 화일 핸들을 DOS에 알려서 그 화일핸들을 DOS에 돌려준다. 즉 화일을 폐쇄한다. 화일의 개방에 따르는 MS-DOS의 처리순서는 다음과 같다.

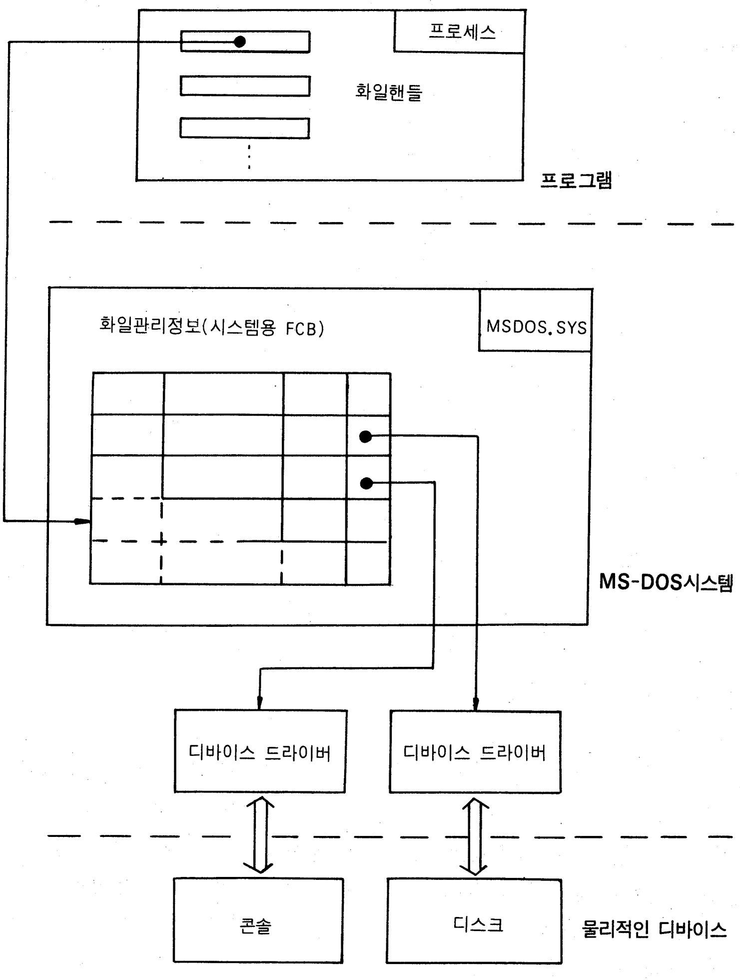
① 시스템용 FCB를 할당하고, 필요한 정보를 설정한다. 링크 카운터를 1로 한다.
② 화일핸들은 시스템용 FCB의 테이블 엔트리를 하나 할당하고, FCB의 번호를 세트한다.
③ 화일핸들을 프로세스에 돌려준다.

또, 화일을 폐쇄할 때의 처리순서는 다음과 같다.

① 지정된 화일핸들은 FCB의 엔트리를 사용하지 않는다고 생각하여 해방한다.
② 지정한 화일핸들에 대응하고 있던 FCB의 링크카운터를 하나 빼고, 0이면 그 FCB를 해방한다.

링크카운터란, 복수의 화일핸들이

〔그림 1〕 화일핸들로부터 디바이스까지의 제어흐름도



하나의 시스템용 FCB를 공유한 경우에 FCB의 내부에 그 FCB를 공유하고 있는 화일핸들의 수를 기억하는 것이다(그림 2 참조). 화일을 폐쇄하는 경우에는 이 링크카운터가 0이 아닌한 FCB는 해방되지 않기 때문에 친프로세스와 자프로세스가 같은 FCB를 매개로 하여 화일을 액세스하고 있을 때, 자프로세스가 화일을 폐쇄하여도 FCB는 해방되지 않으므로 친프로세스의 화일핸들을 정상적으로 이용할 수 있게 되어 있다.

(1) 화일의 개방

화일의 개방은 FN 3DH로 수행된다. 인수로서 개방하는 화일의 화일이름과 화일의 개방모드를 넘긴다. 정상적으로 화일이 개방되면 화일핸들이 돌려진다.

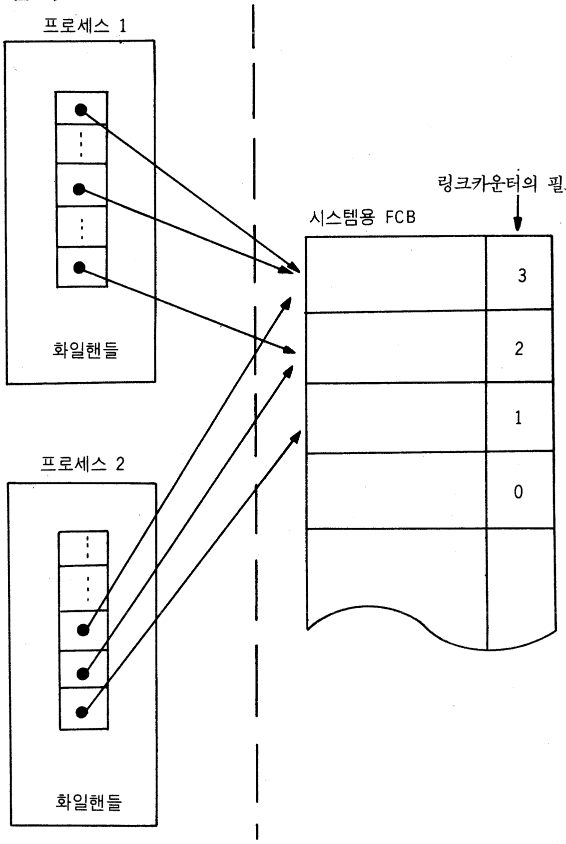
화일이름으로서는 경로명이 붙은 것과 디바이스 명("CON" 및 "PRN" 등)을 지정할 수도 있다. 또, 영문자에 대해서는 대문자로 한정하지 않고, 소문자의 화일이름을 넘기더라도 정상적으로 처리되기 때문에 대부분의 경우에는 명령어라인에서 지정된 화일이름을 그대로 넘길 수 있는 것이다. 또, 화일이름에 와일드 문자를 포함할 수 없기 때문에 와일드 문자를 포함하여 화일이름을 지정한 경우에는 미리 전개해 둘 필요가 있다. 또 화일 이름으로서 디렉토리 이름을 넘기면 당연히 에러가 된다.

화일의 개방모드란 그 화일을 어떤 형식으로 액세스할 것인지(읽기, 쓰기 등)의 구별이다. 즉 화일에 대한 액세스 방법을 알려주는 모드인 것이다. 이들 모드를 지정하는 것에 의해 읽어내기용의 화일에 써넣는다든지, 써넣는 중의 화일에서 읽어내기를 한다든지 하는 틀린 작동을 수행시키는 프로그램의 오류에 의한 화일의 파괴를 방지할 수 있다. 또 써넣기 금지의 화일 속성을 갖는 화일은 읽어내기 전용이외의 모드로 개방할 수 없기 때문에 화일의 보호가 가능하다. 모드로서는 읽어내기 전용, 써넣기 전용, 읽고 쓰기 양용의 3종류를 지정할 수 있다. 또 버전3.1 이후에서는 이들 모드 지정이 확장되어 다른지정도 가능하다.

이 3DH 펑션 콜에서 주의를 해야할 사항은 개방할 수 있는 화일이 기존의 화일 밖에는 허용되지 않는다는 것이다. 즉 개방모드가 "써넣기"라 하여도 그 화일이 존재하지 않으면(즉, 새로이 작성하는 경우에는) 화일의 개방에 실패한다. 화일을 새로이 작성하는 경우에 대해서는 펑션 3CH를 사용한다. 이것에 대해서는 이후에 설명한다.

리스트 1은 펑션 콜에 의한 개방함수

〔그림 2〕 FCB의 링크카운터



〔리스트 1〕 화일 개방함수

```
/*      pcopen.c        */

#include "intdos.h"

int open(fname, mode, pmode)
char *fname;
int mode;
{
        union REGS inregs, outregs;

        inregs.h.ah = 0x3d;
        inregs.h.al = (unsigned char)mode;
        inregs.x.dx = (unsigned int)fname;
        intdos(&inregs, &outregs);
        if (outregs.x.cflag)
            return -1;
        else
            return outregs.x.ax;
}
```

pcopen을 작성한 것이다. 이 함수는 인수로서 화일이름과 개방모드를 지정하여 개방할 수 있으면 그 화일핸들을, 에러이면 "−1"을 돌린다. 개방모드(액세스모드)는 지난달에 소개했던 화일액세스 방법중에서 O_RDONLY, O_WRONLY, O_RDWR 중의 하나를 지정하도록 작성하였다.

(2) 화일의 폐쇄

화일의 폐쇄 펑션 3EH를 사용하여 수행한다. 인수로서 폐쇄하는 화일핸들을 넘기면 그 화일핸들이 폐쇄된다. 리스트 2는 펑션 3EH를 사용하여 작성한 화일 폐쇄함수 pcclose이다. 이 함수는 인수로서 화일핸들을 지정하여 그 화일핸들을 폐쇄할 수 있으면 "0", 에러이면 "−1"을 돌린다.

화일핸들을 이용한 입출력

화일을 읽어내는 것은 펑션 3FH, 화일로 써넣는 것은 펑션 40H에 의해 수행되지만 이들 펑션은 다음과 같은 특징을 갖는다.

- 데이터는 레지스터가 아니라 반드시 메모리안에 놓여진다.
- 데이터 길이는 65536까지의 임의의 길이로 프로그램측에서 제공한다.

C의 입출력 라이브러리 중에서도, 일반적으로 "저수준 입출력"이라고 불리는 read 및 write 등의 함수의 규격은 사실상 이들 펑션 콜과 거의 같은 것이다. 리스트 3에 read 함수를, 리스트 4에 write 함수를 각각 대응하는 펑션 콜에 의해 작성한 예를 보인다.

프로그램 안에서 입출력의 단위로서 많이 이용되는 것은 문자(바이트) 단위, 행 단위, 블록(프로그램이 지정하는 고정 길이) 단위 등 3종류이다. 한편, 펑션 콜이 제공하는 입출력 단위는 블록뿐이기 때문에, 그 이외의 단위로 입출력을 이용하는 경우에는 블록 단위의 입출력을 이용하여 루틴을 작성할 필요가 있다.

C언어에 의한 프로그램을 개발하는 경우에는 각종 단위에 의한 입출력용 라이브러리 함수가 지원되고 있으므로 그들을 이용하는 것만으로도 프로그램의 작성이 가능하다.

화일의 작성과 삭제

이미 앞에서 설명한 것처럼, 화일 개방의 펑션 콜에서는 화일의 작성까지는 수행하지 않기 때문에, 새로운 화일을 작성하는 경우에는 별도의 펑션 콜을 사용할 필요가 있다. 화일 개방 및 폐쇄를 수행하는 펑션 콜은 디스크 화일과 디바이스 화일 양쪽 모두 사용할 수 있지만, 화일을 작성하거나 삭제하는 펑션 콜은 디스크 화일에서만 사용할 수 있다.

〔리스트 2〕 화일 폐쇄함수

```
/*      pcclose.c       */
#include "intdos.h"

int close(fh)
int fh;
{
        union REGS inregs, outregs;

        inregs.h.ah = 0x3e;
        inregs.x.bx = (unsigned int)fh;
        intdos(&inregs, &outregs);
        return -outregs.x.cflag;
}
```

〔리스트 3〕 화일읽기함수

```
/*      pcread.c        */
#include "intdos.h"

int read(fh, buf,nbytes)
int fh;
char *buf;
unsigned int nbytes;
{
        union REGS inregs, outregs;

        inregs.h.ah = 0x3f;
        inregs.x.cx = nbuytes;
        inregs.x.bx = (unsigned int)fh;
        inregs.x.dx = (unsigned int)buf;
        intdos(&inregs, &outregs);
        if(outregs.x.cflag)
                return -1;
        else
                return outregs.x.ax;
}
```

〔리스트 4〕 화일쓰기함수

```
/*      pcwrite.c      */

#include "intdos.h"

int write(fh, buf, nbytes)
int fh;
char *buf;
unsigned int nbytes;
{
        union REGS inregs, outregs;

        inregs.h.ah = 0x40;
        inregs.x.cx = nbytes;
        inregs.x.bx = (unsigned int)fh;
        inregs.x.dx = (unsigned int)buf;
        intdos(&inregs, &outregs);
        if(outregs.x.cflag)
             return -1;
        else
            return outregs.x.ax;
}
```

〔리스트 5〕 화일작성함수

```
/*      pccreate.c      */

#include "intdos.h"

06/29/89efine S_IREAD       (0x00)
06/29/89efine S_IWRITE      (0x01)
06/29/89efine S_IHIDDEN     (0x02)
06/29/89efine S_ISYSTEM     (0x04)

int creat(fname, pmode)
char *fname;
int pmode;
{
      union REGS inregs, outregs;
      inregs.h.ah = 0x3c;
      inregs.x.cx = (unsigned int)(mode^S_IWRITE);
      inregs.x.dx = (unsigned int)fname;
      intdos(&inregs, &outregs);
      if(outregs.x.cflag)
           return -1;
      else
          return outregs.x.ax;
}
```

또 실제로 이용할 때는 이들의 사용방법을 의식하지 않아도 될 수 있도록 화일 개방(open)의 펑션 콜과의 교환을 판단하여 개방(또는 작성)을 수행하는 함수를 준비하여 하나의 함수로 화일 작성과 보통의 개방을 모두 지원하도록 하면 좋을 것이다.

(1) 화일의 작성

화일의 작성은 펑션 3CH에 의해 수행된다. 인수로서는, 작성하는 화일이름과 그 화일의 속성을 넘긴다. 정상적으로 화일이 작성된 경우에는 화일핸들이 돌려진다.

이 펑션 콜로 기존의 화일을 지정하면 그 화일의 속성이 써넣기금지로 되어 있지 않는 한, 그 내용은 지워지므로(즉, 화일의 크기가 0 바이트로 된다) 주의가 필요하다. 또 써넣기 금지 속성이 붙은 기존의 화일을 지정하면 에러가 된다.

이 펑션 콜로 지정하는 속성이란 그 화일에 대한 액세스 모드가 아니라, 그 화일에 붙는(화일 시스템에 기억된다) 속성이다. 돌려지는 화일핸들 그 자체는 지정된 속성에 관계없이 써넣는 것이 가능하게 되어 있지만, 이 후의 개방(open)때에는 그 속성에 따라서 보호가 수행된다. 지정할 수 있는 속성으로서는 써넣기금지/은폐화일/시스템화일의 3종류에 대해서 각각 유효하게 할지 어떨지의 지정을 수행한다.

리스트 5의 프로그램은 펑션 콜을 사용하여 C언어로 작성한 함수 pccreat이다. 인수로서 화일 이름과 그 화일에 붙는 속성을 지정한다.

(2) 화일의 삭제

화일의 삭제는 펑션 41H로 수행한다. 이 펑션 콜은 화일핸들에 의해서가 아니라 경로 이름으로 지정된 화일을 삭제한다. 화일이름 중에 와일드문자가 있는 경우와 써넣기금지 속성이 지정되어 있는 화일을 지정하면 에러가 발생한다.

〔리스트 6〕 화일삭제함수

```
/*      pcunlk      */

#include "intdos.h"

int unlink(fname)
char *fname;
{
     union REGS inregs, outregs;

     inregs.h.ah = 0x41;
     inregs.x.dx = (unsigned int)fname;
     intdos(&inregs, &outregs);
     return -outregs.x.cflag;
}
```

개방(open) 되어 있는 화일을 삭제하는 것은 가능하지만(MS-DOS에서는 보호되어 있지 않는다), 삭제 후에 화일을 액세스하면 에러가 발생한다. 특히 화일로의 써넣기를 실행하면 디스크의 내용이 파괴되어 치명적인 결과를 초래하는 경우도 있다.

또 버전3.1 이후의 MS-DOS에서는 SHARE 명령어를 MS-DOS에 부속하여 이와 같은 에러를 방지할 수 있도록 되어 있다. 리스트 6에 표본으로서, 화일을 삭제하는 함수 pcunlk를 작성하였다. PC

마이크로프로세서 마이크로컴퓨터 (하드웨어 과정) 講義案內

개강 매월 5일 (2개월 완성)

본원에서는 금번 2개월 과정으로 digital circuit 설계로 부터 CPU 및 memory, 주변장치 등의 구조 및 동작원리를 습득하여 computer system 및 주변기기의 Interface를 설계하는 능력을 함양하고자 합니다. 또 이 과정에서는 Z-80계 computer 및 APPLE, IBM PC 등의 system을 대상으로 구성, 동작원리, 프로그래밍 실습을 하여 궁극적으로 MICOM 및 digital system의 설계 및 수리능력을 배양하고자 하오니 많은 참여 바랍니다.

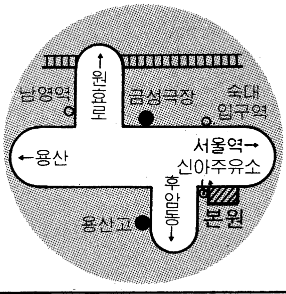
■ 강습일정

주별	일차	강습 내용 (이론 및 실기)	주별	일차	강습 내용 (이론 및 실기)
1주	1일차	computer의 기본 구조와 8bit CPU의 종류 및 내부 구성	5주	1일차	Z80 CPU의 구조 및 Timing 특성
	2일차	논리회로 설계원리 및 그 응용		2일차	Z80 CPU의 Interrupt (IM0, IM1, IM2)
	3일차	마이크로 컴퓨터 시스템의 구성		3일차	μ-Computer의 회로설계 및 실습
2주	1일차	Karnaugh Map을 이용한 디지탈회로 설계	6주	1일차	Assembly Language를 이용한 프로그램 실습
	2일차	Flip Flop 및 Counter 등을 이용한 응용회로 설계		2일차	8255 및 Z80 P10의 특성 및 응용
	3일차	CPU의 주변회로 및 Address Map/Decoding		3일차	16bit MPU(MC68000)의 구조 및 원리
3주	1일차	TTL 논리회로 설계 실습	7주	1일차	Z80 S10의 특성 및 응용
	2일차	A/D 및 D/A Converter 및 ALU		2일차	EPROM Writer 및 Floppy Disk Controller
	3일차	Memory(ROM/RAM/DRAM)의 구성		3일차	MC68000의 Signal 및 Bus Operation
4주	1일차	8080 및 8085 CPU의 구조 및 Timing 특성	8주	1일차	음성합성 장치의 설계
	2일차	8085 Assembly언어 프로그래밍		2일차	Bus Standard, Modem, Data통신 소개
	3일차	I/O(Port, Keyboard etc)의 원리 및 Video Generator		3일차	16bit MPU(MC68000)의 회로 설계

Textbook

1. TTL응용실무
2. Z80 마이크로컴퓨터
3. Microprocessor Interfacing Technique
4. Microprocessor and Digital Systems

Hardware + Software = ?



전자산업 4반세기를 지켜온 기술인의 전당

과인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

입출력 운영방식

김용득 / 아주대 전자과 교수

입출력 접속이론

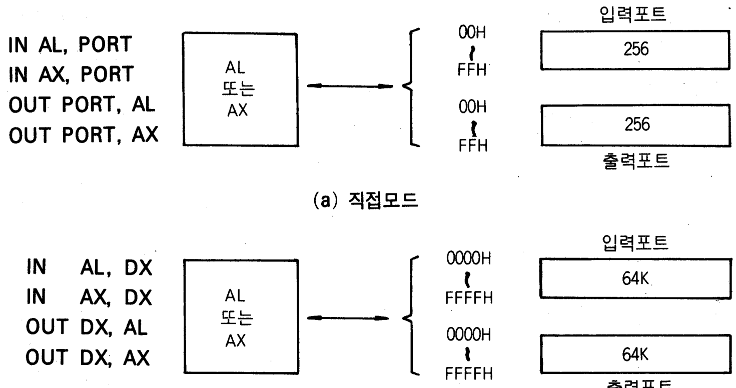
8086/286을 사용한 마이크로 프로세서 시스템에서 입출력 장치와의 정보 전송방식은 입출력 기기에 따라 8비트 또는 16비트 정보 모두가 가능하다. 프로세서로 입출력하기 위해서는 지난 6월호에서 언급한 바와 같이 기억소자에 입출력장치를 할당하여 사용하는 메모리 맵 I/O (Memory mapped I/O)방식과 기억소자와 별도의 입출력 영역을 지정하여 접속시키는 분리 I/O (Isolated I/O) 즉 포트 입출력 방식으로 설계 가능하다. 그러나 8086/286에 관련된 프로세서 시스템에서는 포트방식에서도 6만4천개의 입출력 장치 접속이 가능하므로 이 방식만 사용되고 있다.

또한 포트 입출력 접속방식에서도 입출력장치가 각각 256개 미만일 때는 직접 주소방식을 사용하고, 입출력장치가 256개 이상일 때는 간접주소방식(6400개까지 지정가능)을 사용하게 된다. 이때 포트방식이 사용될 경우에 반드시 프로세서의 내부 레지스터는 AL(입출력 기기가 8비트인 경우), AX(16비트인 경우)가 사용되며 포트지정을 위하여 직접 명령을 수행하게 된다.

이를 요약하면 **그림 1**과 같이 IN 또는 OUT명령어로 외부장치와 정보전송이 이루어짐을 볼 수 있다. 이를 하드웨어로 구성할 때 즉 IN이나 OUT 명령어가 수행처리될 때는 $\overline{RD}$, $\overline{WR}$ 및 IO/$\overline{M}$ 제어신호와 DX값 또는 포트 주소값이 주소선에 나타나도록 프로세서에 의해 발생되므로 이들 신호를 사용하여 외부 입출력장치의 작동회로를 구성하게 된다.

그림 2는 입출력장치를 프로세서시스템에 접속하기 위한 기본설계방식이며 아무리 복잡한 입출력장치의 접속회로도 항상 이와같이 구성된다. 즉, 입

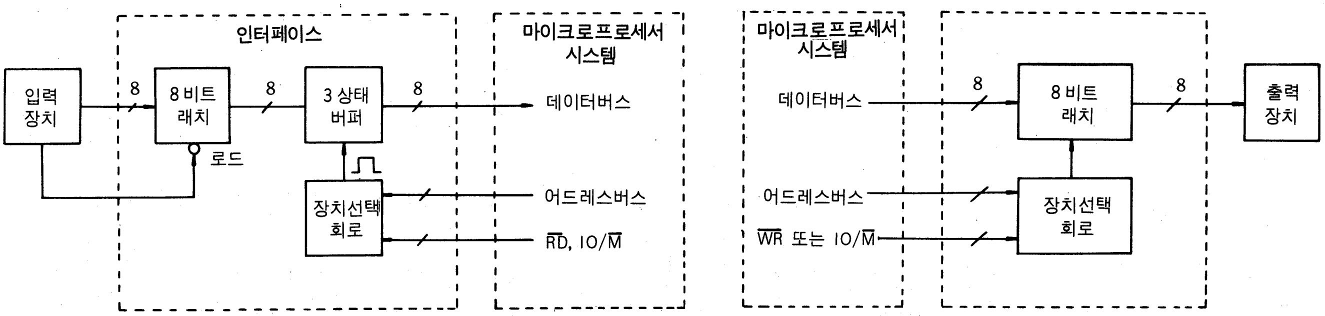
〔그림 1〕 입출력 명령어 방식



(a) 직접모드

(b) 간접모드

〔그림 2〕 입출력장치의 설계



(a) 입력장치의 설계

(b) 출력장치의 설계

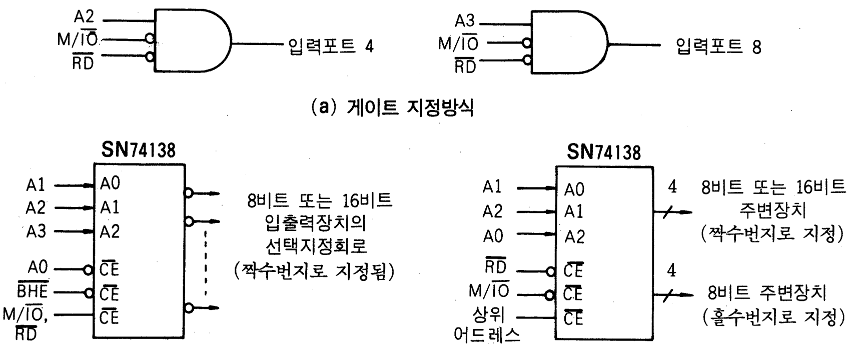
력장치에 대한 접속회로는 그림 2 (a)와 같고, 마이크로 프로세서 시스템에 대해 시간과는 무관하게 작동상태가 발생한다. 따라서 발생된 이들 정보값을 프로세서가 내부 작동시간에 동기시켜 처리하기 위하여 레지스터에 저장하여야 한다. 그러면 프로세서는 이들 레지스터에 저장된 정보가 새로 입력된 것인지 또는 이미 처리된 정보인지를 검사하여 시스템 내부로 입력된 값을 읽어들이게 되는 것이다. 마찬가지로 출력장치의 접속모드에서도 출력장치에서 레지스터에 저장된 정보를 읽어 처리했는가를 확인하여 새로운 값을 프로세서 내부 레지스터 (AL 또는 AX) 로부터 접속회로의 버퍼에 기억시키게 된다.

이상과 같이 입출력장치의 접속회로란 버퍼에 기록하고 읽는 과정이 시간차를 두고 작동이 이루어지도록 하는 것으로서 레지스터에 정보를 기억시키게 하기 위해서는 로드 (Load) 단자에 신호를 인가해 주면 되고, 읽기 위해서는 레지스터 출력에 연결된 3상태 출력단자 (3-state Output)를 "ON" 시키면 된다. 이들 신호는 입출력장치 쪽에서는 비교적 단순하게 발생되나 마이크로프로세서부에서는 제어신호 ($\overline{RD}$, $\overline{WR}$, IO/$\overline{M}$) 가 단순한 경우 **그림 3(a)**와 같이 게이트로서 구성 (Linear Selection Method라 부름) 될 수 있다. 그러나 입력포트가 여러개인 경우 프로그램상에서 특별한 주의를 요한다.

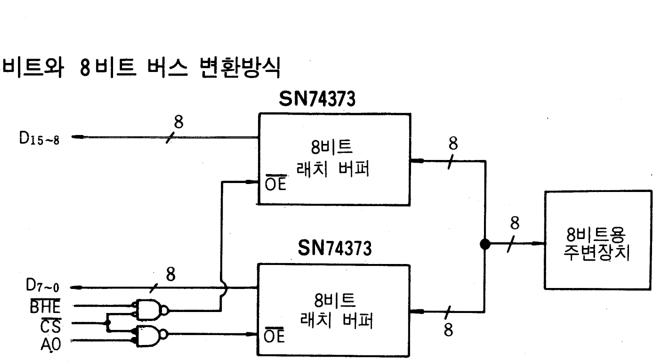
즉, 그림 3(a)에서와 같이 게이트로서 4번 포트와 8번 포트를 설계할 경우 "IN AL, 04H" 또는 "IN AL, 08H"로 사용하는 경우에는 전혀 문제가 없으나 "IN AL, 0CH"로 사용하는 경우 4번과 8번 포트에 연결된 입력장치는 동시에 호출되어 하드웨어가 손상된다. 따라서 이러한 문제를 해결하기 위해서 그림 3(b)와 같은 디코더를 사용한 방식이 보편적으로 사용되고 있다. 그림 3 (b)는 8비트 또는 16비트 입출력장치인 경우의 접속방식을 보여준 것으로 $\overline{BHE}$와 A0 (LSB주소선) 로서 8비트 또는 16비트가 지정된다. 그러나 16비트 입출력장치를 직접 접속하지 않고 8비트 입출력장치를 부착하여 내부 비트로 변환하여 사용하고자 할때는 **그림 4**와 같이 구성하면 된다.

이들 동작상태를 IBM-PC에서 예를 들어 설명하면 **그림 5**와 같다. 사용자 프로그램에서 입출력장치를 사용하고자 할때는 입출력장치를 처리해주는 모니터 프로그램을 인터럽트식으로 지정하여 이루어지게 한다.

〔그림 3〕 포트지정신호의 발생회로



〔그림 4〕 16비트와 8비트 버스 변환방식



MSI 접속소자

지금까지는 입출력 포트의 접속원리를 설명하였으나, 이제부터는 지금까지 설명된 접속회로를 구성하는데 칩 (MSI와 LSI)을 어떻게 사용하는가에 대해 논의하기로 하겠다.

먼저 입력 또는 출력장치에 사용되는 기본회로는 그림 5와 같이 소스 (Source) 시스템에서 목적 (Destination) 시스템으로 정보가 이동하는 것으로 요약될 수 있다. 즉, 입력방식일때 소스시스템은 입력장치, 목적시스템은 컴퓨터가 되고, 출력방식일때 소스시스템은 컴퓨터, 목적시스템은 출력장치로 할당된다. 이들 소스 또는 목적시스템은 각각의 동작클럭에 의하여 작동되고 접속회로에서 이들 정보를 시간적으로 연결시켜 주게 된다. 따라서 이들 목적에 적합하도록 MSI소자와 LSI소자가 이미 상용화되어 있으므로 이들을 소개하고, 설계 및 운용과정을 설명하도록 해본다.

SN74373 또는 SN74374라는 접속소자는 **그림 6(a)**와 같이 플립플롭과 3상태 출력부 8개가 병렬로 하나의 IC에 연결되어 있으므로 8비트 정보를 래치에 입력시키기 위해서는 클럭 (CLK)을 인가하면 기억되고, 또 플립플롭에 기억된 정보를 출력선에 접속하기 위해서는 출력제어 (OC) 단자를 사용한다. 이들 두개의 제어신호 즉, CLK와 $\overline{OC}$

신호는 소스시스템과 목적시스템에 따라 임의의 시간에 작동되며 이들을 시간적으로 일치시키는 것에 대해서는 동작모드에서 논의하기로 한다.

소스와 목적시스템에서 시간적으로 발생시간이 다를 경우 이와 같은 SN74373(SN74573도 같은 종류임) 칩을 사용하지만 두 시스템의 동기된 클럭에 따라 동시에 작동되는 동기식 제어방식인 경우에는 SN74373칩에서 플립플롭이 생략되고 단지 3상태버퍼만 갖는SN74240 또는 SN74244 등이 사용된다.

그리고 이들 MSI소자는 한 방향으로만 사용 가능하나 때에 따라서는 하나의 하드웨어로 입출력모드가 동시에 사용될 경우가 있으며(예를 들면 디스크장치 : FDD 또는 HDD는 똑같은 하드웨어 시스템 버스를 사용하여 정보를 읽고, 기록함), 이 경우에는 트랜시버라 부르는 SN74245 (인텔계열에서는 8286, 82286)란 소자를 사용한다. 이들 트랜시버 1비트를 블록으로 나타내면 **그림 6**(b)와 같으며 제어신호에 따라 A시스템에서 B로, 또는B시스템에서 A로 정보이동이 가능하다.

지금까지 논의된 이들 소자는 모두 8비트인 경우지만 때에 따라서는 LED 제어 및 로보트 등에서 리미트(Limit) 센서 등과 같이 입출력장치가 1비트로 구성된 경우가 있다. 이 경우 효율적 운영을 위하여 하나의 소자에 여러개의 입출력장치를 접속시킬 수가 있다. 즉, **그림 7**(a)와 같이 내셔널사의 9334라는 소자는 주소 선택단자에 따라 1비트 출력장치를 8개나 접속 가능하므로 하나의 IC로서 8대 모터의 구동을 각각ON / OFF시키는 제어등에 사용된다.

또한 **그림 7**(b)는 인텔계열 소자로서 1비트로 구성된 입력장치를 16대까지 접속 가능하고 버스선의 특정비트를 사용하여 각 장치의 입력값을 읽어 들일 수가 있다.

이와 같은 특수한 목적의 접속소자는 많은 종류가 개발되어 있으므로 데이터북에서 참조하기 바란다.

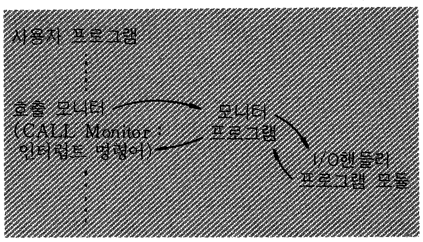
입출력 운영방식

앞에서 설명한 소자로 하드웨어가 설계되면 이를 정확하고 효율적으로 작동시키기 위해서 소프트웨어로 처리하게 된다. 이러한 일련의 과정이 외부 입출력장치와 마이크로프로세서 시스템간의 정보전송에 적용되면 다음의 3가지 방법으로 요약된다. 즉, 소프트웨어로 정보의 입출력 상태를 검지하여 처리하는 프로그램제어형 입출력방식과 하드웨어에 의하여 새로운 정보상태의 입출력을 처리하기 위한 인터럽트방식 및 대단위 정보를 마이크로프로세서 개입없이 입출력장치로부터 직접 시스템의 기억장치에 정보를 이동시킬 수 있는 DMA(Direct Memory Access)방식으로 구분된다. 따라서 본 절에서는 이들 3가지 방법을 세부적으로 설명하기로 한다.

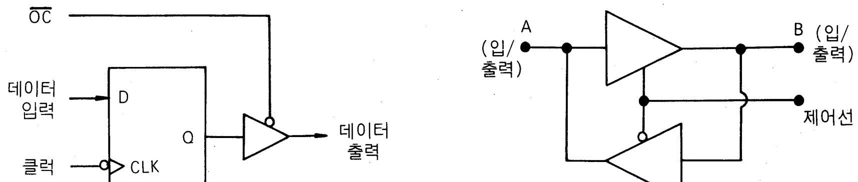
먼저 프로그램제어형 입출력방식에는 무조건적 처리방식(Strobe방식)과 조건부 처리방식(Handshake방식)으로 구분된다. 무조건적 처리방식은 BCD 표시장치에 숫자를 표시하기 위한 것과 같이 사용될 입출력장치의 준비상태와는 무관하게 마이크로프로세서에 의해 일방적으로 정보를 처리하는 방식으로 특수 목적 이외에는 사용하기가 어렵다. 이러한 문제점을 보완한 것으로 핸드세이킹 입출력방식이 사용되는데 이 방식은 정보를 전송하기 전에 입출력준비상태(이를 Flag 또는 Status라 부름)를 검사하여 준비되면 입출력시키는 방식이다. 즉, **그림 8**에서와 같이 입출력장치의 준비상태가 완료될 때까지 CPU검사를 계속하다가 준비가 끝나면 정보를 처리하는 방식으로 저속의 입출력장치에 동작시간을 맞추어 CPU를구동시키는 방식인데 CPU처리시간을 낭비하므로 그다지 효율적인 것은 아니라고 본다.

이를 입력장치(키보드라 가정)에 적용해서 실례를 들고, 하드웨어를 설계

〔그림 5〕 IBM-PC의 입출력동작 흐름도

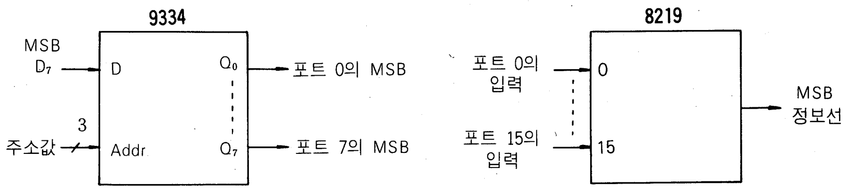


〔그림 6〕



(a) 1비트 래치 버퍼회로 (b) 트랜시버 등가회로

〔그림 7〕 1비트 입출력소자의 접속방식

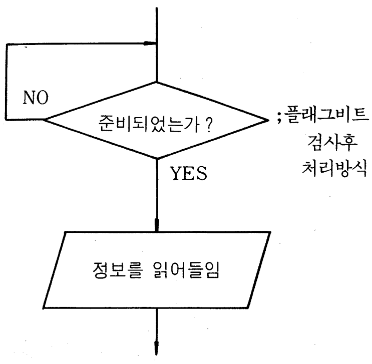


(a) 1비트 출력포트 (b) 1비트 입력포트

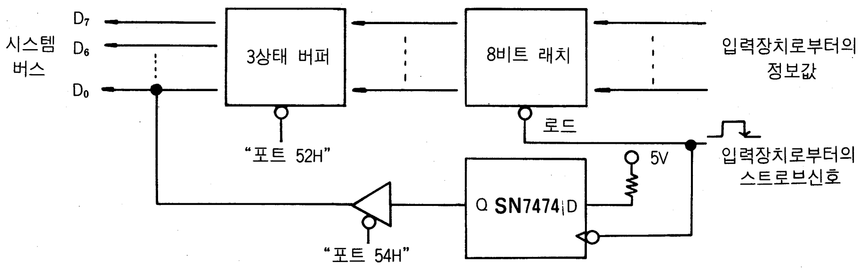
하여 동작 운용프로그램을 작성하여 보도록 한다. 키보드에서 키를 누르면 키를 눌렀다는 신호(Strobe)가 발생되며, 이 신호에 따라 그 키에 해당하는 코드값(ASCII코드)이 8비트 래치에 기억된다. 물론 이때 이 Strobe신호도 저장시켜야 하며 이는 키가 눌렀다는 즉, 새로운 키값이 입력되었음을 마이크로프로세서 시스템에 알려주는 상태(Status)신호로 해석된다. 따라서 **그림 9(a)**의 하드웨어에서는 키 입력보드에 대하여 입출력포트를 2개로 할당시켜(정보포트와 스테이터스포트) 이를 54H, 52H로 지정하도록 하였으며 54번 포트로부터 D_0버스선에 연결된 1비트값을 읽어 이를 검사하므로 래치회로의 키 입력상태가 새로 들어왔음을 알려준다. 이를 핸드세이킹방식으로 확인하여 입력시키는 운용과정을 **그림 9(b)**에서 프로그램으로 보여주고 있다.

이들 핸드세이킹방식으로 입력장치에서 정보를 읽거나 쓰는 경우 이들 장치가 준비되었는가를 확인하여야 하며 이때 프로세서가 검사하는시간은 약10 μsec정도이고, 입력장치인 경우 동작시간은 즉, 키보드라 가정할때 초당 10자 입력이 표준이므로 이 경우1바이트 입력에 소요되는 시간은 10만μsec 가 된다. 따라서 한 글자를 받아들이기 위해서 프로세서는 검사시간 10μsec 를 뺀 나머지인 99990μsec동안 입력장치의 준비상태를 확인하기 위해서 대기하여야 하는 비효율적인 운용이 예상된다. 따라서 이러한 문제를 해결한 입출력방식이 인터럽트방식이며, 동작방식은 입출력장치의 동작과는 무관하게 프로세서는 수행되는 주프로그램을 처리한다. 그러나 이때 입출력장치에서 정

〔그림 8〕 핸드세이킹 입출력방식



〔그림 9〕 핸드세이킹 입력장치의 설계

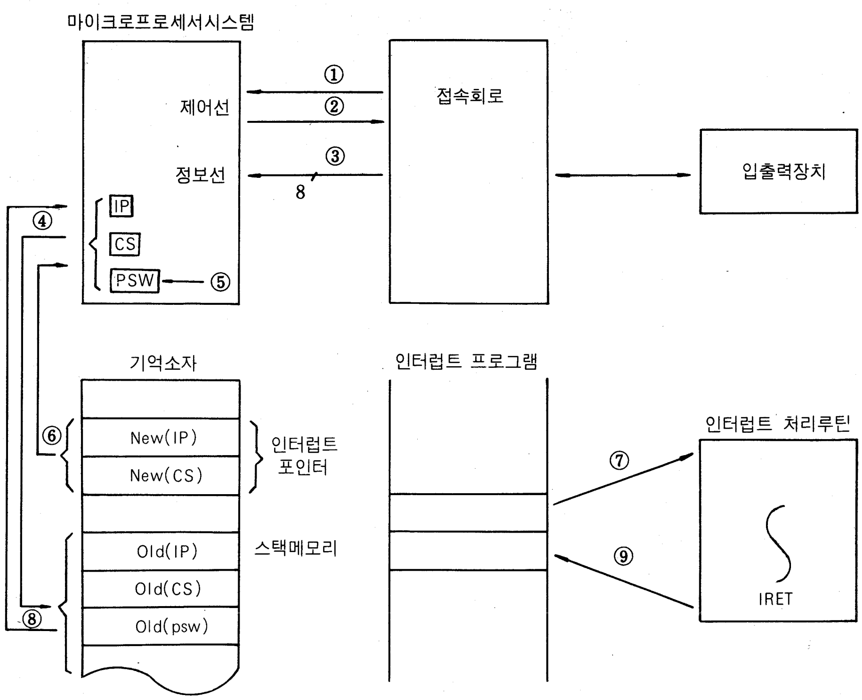


(a) 하드웨어의 구성

```
TEST : IN    AL, 54H        ;54번 포트로부터 status값을 읽어들임
       TEST AL, 01H         ;status비트 검사
       JZ    TEST
       IN    AL, 52H        ;52번 정보값을 읽어들임
```

(b) 작동프로그램 예

〔그림 10〕 인터럽트 처리과정



① 인터럽트 요청신호(INTR)
② 인터럽트 인지신호(INTA)
③ 벡터주소의 지정
④ 스택메모리에 IP, CS, PSW저장
⑤ IF 및 TF 플래그비트 클리어
⑥ 인터럽트 포인터로부터 서비스루틴의 지정
⑦ 인터럽트 처리과정의 수행
⑧ IRET명령어에 따라 스택에 저장된 IP, CS값 재기억
⑨ 주프로그램으로 환원

보전송이 필요한 경우는 비기동적으로 임의 시간에 하드웨어적인 신호를 프로세서로 보내 자료처리를 요구하게 된다. 그러면 프로세서는 이들 요청된 하드웨어에 따라 프로세서에 내재된 특정번지의 처리 프로그램을 처리하게 되는데, 이를 벡터 인터럽트(Vectored Interrupt) 방식이라 한다.

8086/286 프로세서 시스템에서 인터럽트는 256개가 있으며 세부적 사항은 다음호의 인터럽트 처리방안에서 논의하기로 하고 본란에서는 이들 기본동작만 **그림 10**을 보면서 설명하도록 하겠다. 입출력장치에서 키입력이 되었거나 또는 출력시킬 준비가 되어 있으면 입출력장치로부터 프로세서에게 준비된 입출력장치를 처리해 달라는 신호(INTR : 인터럽트 요청신호)를 보내게 되며, 이 신호를 인지한 프로세서는 처리하던 프로그램을 일시 중단하고 입출력장치를 처리해 준다는 신호를 발송한다(INTA : 인터럽트 인지신호). 그러면 입출력장치에서는 INTA신호를 수신하여 어느 인터럽트 방식을 사용할 것인가에 따라 정의된 벡터 주소값을 전송하게 된다.

프로세서는 현재 처리하고 있던 프로그램번지 값인 IP, CS, PSW를 스택 메모리에 저장하고 인터럽트 벡터로부터 새로 처리될 주기억장치의 번지를 읽어와 해당 입출력 프로그램을 처리한 IRET명령을 만나면 해당 루틴이 끝나고 기억된 주프로그램의 IP, CS값으로 되돌아와 주프로그램을 계속 처리한다. 이와같은 하드웨어 방식에 의한 인터럽트 입출력방식은 LSI 접속소자를 설명할 때 IBM-PC에 연관시켜 다시 세부적으로 설명하기로 한다.

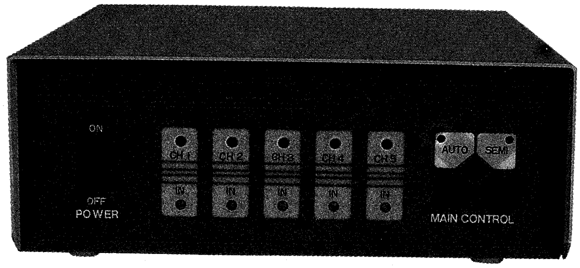
핸드세이킹이나 인터럽트에 의한 입출력장치와의 정보전송은 반드시 프로세서가 중재되고 프로세서의 내부레지스터를 사용하여 기억소자와의 정보이동이 이루어지게 된다. 그러나 프로세서가 중재되지 않고 고속으로 기억소자와 입출력장치와의 정보전송이 이루어지기 위해서는 DMA방식이 사용되며, 하드디스크 장치로부터 정보를 기억소자에 이동시키기 위한 고속처리에 사용하고 있다. 이 방식도 LSI 접속회로 설계에서 세부적으로 설명하기로 하고 본론에서는 생략한다.

프로세서의 하드웨어 시스템을 설계한다고 하면 이들 LSI 접속소자의 동작상태를 이해하는게 가장 중요하다. 따라서 본 강좌에서도 이를 중점 강조하여 다음호부터 세부적으로 설명하기로 하겠다. PC

PRINTER 주변기기 전문 메이커

- ●컴퓨터는 2대 혹은 3대이고, 프린터는 1대뿐이라 불편하게 사용하고 계십니까?
- ●컴퓨터 1대에 프린터를 1대씩 사용하거나 추가 구입시 프린터를 구매하려고 생각하십니까?
- ●프린터를 하고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- ●CAD 사용시 Plotter가 그리고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- ●컴퓨터의 수량과 프린터의 수량이 안맞아 적절한 사용을 못하고 계십니까?
- ●학교 및 학원의 실습실에 프린터대수가 부족해 적절한 교육을 못하고 계십니까?
- ●F.A 혹은 계측기쪽의 프린터 인터페이스에 문제가 있읍니까?

컴퓨터 사용중에 불편한 위의 모든 문제를 자신있게 처리해 드리므로, 귀사의 업무에 도움이 되어 드리고 싶읍니다!



※ K-P-345A는 컴퓨터 프로그램 보호법에 의해 등록된 제품입니다.

K-P-345A의 사양

입·출력 방식	Centronics Type
입력 COMPUTER	3 : 1~5 : 1
출력 PRINTER	1 대
대기 시간	5초, 10초, 20초, 40초 중 선택
채널 변환방식	전자식
채널 선택방식	자동 or 반자동
용지 F.F 기능	자동 F.F or None F.F
F.F 삭제 기능	자동 삭제 or None

K-256P의 사양

버퍼 용량	256kB(80col 약 200장)
입·출력 방식	Centronics Type
입·출력 수	1 : 1 or 2 : 1
사용자 MODE	10EA
고유 기능	Data 중 Space 입력 기능, 재 Copy 기능, 메모리 기능, Self Test 기능 등 총 10가지



● DATA SWITCH

- 전자식 완전자동 (2 : 1~8 : 1 or 64 : 1)
- 전자식 반자동 (2 : 1~8 : 1, 4 : 2)
- Serial 완전자동(4 : 1, 8 : 1)

● DATA BUFFER

- 소용량 64kB
- 고성능 다기능 256kB
- 대용량 다기능 1MB
- Serial용 1MB

● DATA CONVERTOR

- Serial→PARALLEL
- PARALLEL→Serial

● 각급 학교 및 학원의 전산 실습실 프린터 공유장치 공사

ATTA SYSTEM
PRINTER 주변기기 전문메이커
서울 · 종로구 종로 3가 175-4 (현대APT1309호)
전 화 : 2 7 7 — 1 7 3 9
F A X : 2 7 7 — 8 2 5 1

●상기 제품은 아래의 전문 취급점에서 전시 판매하고 있으며 직접 사용해 보실 수 있읍니다.●

서울 : 세운상가 협진 274—5419(가동 라열 426호)
용산 나진상가 동해전자 719—8010(17동 가열 304호)
용산 전자랜드 협진 712—2764(3층 C-336호)

인천:바이너리 컴퓨터 (032)762—5544
대구:범진 컴퓨터 (053)252—2211
마산:한길 컴퓨터 (0551)94—5555
원주:중부 씨스템 (0371)43—7979
부산:한국컴퓨터씨스템 (051) 42—2048

(기타지역 전문 취급점 모집중)

그래픽 화면을 다양한 형태로 하드카피해주는
알라딘의 마술램프

김원준 / 건국대학교 전자과

그래픽에 조금이라도 관심이 있는 사람이라면 화면상에 나타나는 그림 등을 프린트로 출력하여 보고 싶은 충동을 느꼈을 것이다. 그러나 게임화면 등은 프린트로 출력하기가 그리 쉽지 않다. 여기에 소개하는 프로그램은 그러한 어려운 문제를 말끔히 해결해 준다. 그래픽 화면을 16가지의 다양한 형태로 출력해주는 알라딘의 마술램프와 같은 프로그램이다.

우리는 컴퓨터를 사용하면서 그래픽을 자주 접하게 된다. 날이 갈수록 각종 프로그램은 멋진 그래픽을 사용하여 사용자를 사로 잡고 있다. 또, 게임들은 대부분이 그래픽 모드를 사용하며, 우리가 직접 프로그램을 작성할 때 화면에 깔끔하고 잘 정리된 그래프를 그리기 위해서도 그래픽을 자주 사용하게 된다.

더우기 한글카드가 없는 기종에서는 한글을 구현하기 위하여 그래픽 모드에서 한글을 그려주는 방식을 사용한다. 그래픽 모드를 사용하는 한글은 비록 속도는 느리다고 하더라도 지금까지 개발된 한글카드로는 하기 힘든 글자의 확대, 축소, 반전 등의 특수한 효과나 그래픽 문자, 다른 나라의 글자까지도 구현이 가능하다. 그런데 컴퓨터 화면에 멋진 그림이 그려져 있을 때면 그 모습을 어디엔가 보존하고 싶을 때가 있다. 화면을 화일로 만들어 저장할 수도 있지만 가장 보편적인 방법은 프린터로 찍어 두는 것이다. 그래픽 화면을 화면에 나타난 그대로 프린터로 찍어 내는 것을 흔히 '하드카피'라고 부른다.

그래픽에 조금이라도 관심이 있는 독자라면 한번 쯤은 어떤 프로그램을 이용하여 그래픽 화면을 하드카피 해 본 경험이 있을 것이다. 매우 많은 사람들이 사용하는 닥터할로우나 여러 그래픽 패키지들은 대부분 하드카피를 지원한다. 또한 게임 화면을 하드카피해 주는 프로그램들이나 도스에 포함된 GRAPHICS나 HGRAPH와 같은 하드카피 프로그램은 어렵지 않게 구할 수 있다. 하지만 이들 프로그램을 사용하다 보면 몇가지 아쉬운 점을 발견하게 된다. 우선 GRAPHICS, HGRAPH와 같은 프로그램은 화면을 90도 회전시켜서 프린터 용지 한장에 꽉 차게, 혹은 더 큰 크기로 찍어 준다. 때로는 이렇게 찍는 것이 좋을 수도 있지만 대부분의 경우는 닥터할로우에서와 같이 가로로 보기 좋은 크기로 찍어주는 쪽이 편리하다. 이들 프로그램이 화면을 90도 회전시켜서 찍는 것은, 화면과 프린터 핀의 구조에 의해 회전시키는 편이 프로그램 작성에 훨씬 용이하기 때문이다.

우리나라에서 가장 많이 사용되는 그래픽카드인 허큘리스 그래픽카드는 일반적으로 2가지 모드를 가장 많이 사용하고 있다. 한가지는 허큘리스의 표준모드인 720×348의 해상도를 가지는 모드이고, 또 한가지는 한글 프로그램에서 주로 사용하는 640×400 모드이다. 그밖에도 MHP 등에서 사용하는 특수한 모드가 있지만 그 비율은 매우 적다. 게다가 허큘리스는 그래픽 화면으로 2페이지를 사용할 수 있다.

그러므로 우리 실정에 맞는 하드카피 프로그램이라면 이들 모드를 모두 지원해야 할 것이다. 그런데 대부분의 하드카피 프로그램들은 몇가지 기능만을 지원하므로 하드카피 프로그램으로 화면을 찍을 때 모드가 다르면 이상한 모양으로 찍히게 된다.

그런데 필자가 가장 중요하게 생각한 것은 다른 프로그램에서의 사용이었다. 이미 개발되어 있는 많은 하드카피 프로그램은 거의 모두가 램상주 형태이다. 그러므로 우선 하드카피 프로그램을 실행시킨 후 그래픽 프로그램을 돌리고, 화면에 원하는 그래픽이 나타났을 때 특정한 핫(Hot)키를 눌러서 찍어내는 방식을 사용한다.

그러므로 다른 프로그램에서 하드카피루틴을 사용하려면 사용자가 직접 손으로 키를 눌러줘야만 한다. 이것은 프로그램 작성시에 매우 불편한 요소가 되므로, 이러한 경우는 대부분 하드카피 루틴을 직접 작성하게 된다. 이 작업은 매우 번거로운 일일 것이다.

그래서 720×348, 640×400 모드의 페이지 0, 1의 모든 조합을 가로로 작게, 혹은 세로로 크게 두가지 크기로 하드카피 해주며, 화면을 반전시켜서 찍을 수도 있고, 타 언어에서 외부 프로세서 호출방식을 사용할 수 있도록 직접 수행기능을 가지며, 또 BIOS 인터럽트에 하드카피 서비스를 추가하여 대부분의 언어와 다른 프로그램들에서 간단한 명령들로 자유롭게 다양한 모드의 화면을 하드카피 할 수 있도록

해주는 프로그램을 만들어 보았다. 이 프로그램을 사용하면 일반적인 그래픽 화면이나 그래픽 한글화면을 보이는 그대로 인쇄해 낼 수 있게 된다.

사용 가능한 기종

이 프로그램은 가장 많은 보급율을 가지는 허클리스 그래픽카드와 엡슨 호환프린터를 기준으로 한다. 이들과 그래픽 모드에서 호환을 갖는 그래픽카드와 프린터라면 사용 가능하다. 프로그램의 소스를 조금 수정하면 다른 기종에서도 사용 가능할 것이다. 또한 컴퓨터는 MS-DOS를 사용하는 IBM-PC / XT, AT호환(286, 386) 등에서 모두 사용 가능하다.

활용 방법

이 프로그램은 2가지 형태로 사용할 수 있다. 한가지는 일반적인 하드카피 프로그램과 마찬가지로 핫키를 사용하여, 원하는 시기에 직접 키를 입력하여 찍는 방식이고, 또 한가지는 다른 언어나 다른 프로그램 내에서 BIOS 인터럽트나 부프로그램 수행방식을 사용하여, 소프트웨어적으로 제어하는 방식이다. 이들 2가지 방법을 자세히 적어 보겠다.

(1) 핫키로 찍는 법(램상주 형태)

우선 DOS 프롬프트상에서 하드카피 프로그램을 실행시킨다.

화일명 뒤에는 옵션을 사용할 수 있는데 그 내용은 다음과 같다.

① HARDCOPY / 0 : 720✻348 모드 0페이지, 작게 인쇄(가로로)
② HARDCOPY / 1 : 640✻400 모드 0페이지, 작게 인쇄(가로로)
③ HARDCOPY / 2 : 720✻348 모드 1페이지, 작게 인쇄(가로로)
④ HARDCOPY / 3 : 640✻400 모드 1페이지, 작게 인쇄(가로로)
⑤ HARDCOPY / 8 : 720✻348 모드 0페이지, 크게 인쇄(세로로)
⑥ HARDCOPY / 9 : 640✻400 모드 0페이지, 크게 인쇄(세로로)
⑦ HARDCOPY / A : 720✻348 모드 1페이지, 크게 인쇄(세로로)
⑧ HARDCOPY / B : 640✻400 모드 1페이지, 크게 인쇄(세로로)

옵션 4~7, C~F도 사용 가능하지만 각각 0~3, 8~B와 같은 기능을 가지며 대소문자의 구별을 하지 않는다. 옵션을 지시하지 않으면 자동으로 옵션이 '0'인 것으로 인식한다. 사용 중에 옵션을 바꾸고 싶을 때는 다시 실행시키면 된다. 그러면 프로그램은 이미 프로그램이 수행되고 있으며, 파라미터가 변경되었다는 것을 알려준다. 그러므로 여러번 반복적으로 실행시켜도 관계가 없다. 또한 옵션 사용법을 잊었을 때는 엉터리 옵션을 주고 실행시키면 사용법을 자동으로 알려주게 된다.

일단 실행이 되면 프로그램은 램에 상주하게 된다. 그 다음에 원하는 그래픽 프로그램을 돌리고 화면에 그림이 나타나면 프린터를 켜고, CTRL-'+'(오른쪽 키패드에 있는 것)을 눌러 준다. 그러면 화면이 그대로 프린터로 보기 좋게 출력될 것이다. 그런데 화면의 흰점은 프린터에서는 검은 색으로 찍히게 되므로 때로는 색이 반전되는 것이 좋을 때도 있을 것이다. 이때는 CTRL-'−'를 사용한다. 그러므로 CTRL-'+'는 정상, CTRL-'-'는 반전된 화상으로 찍히게 된다.

옵션을 선택하는 예를 들어 보자. NKP, THP 등의 한글화면은 640✻400 모드 0페이지를 사용하므로 옵션을 '1'로 하면 된다. 또한 한글도깨비나 프린터 마스터 등은 640✻400 모드 1페이지를 사용하므로 옵션'3'이 된다. 닥터할로우나 orCAD 등은 720✻348 0페이지인 표준모드를 사용하므로 옵션 '0' 혹은 옵션을 주지 않으면 된다. 그리고 이들을 크게 인쇄하고 싶을 때는 각 옵션에 8을 더해 주면 된다.

(2) 각종 언어나 다른 프로그램에서 사용하는 방법

IBM-PC는 DOS나 BIOS 등에 들어 있는 유용한 서브루틴들을 인터럽트 형태로 제공한다. 이 프로그램은 그중 프린터 처리에 사용되는 BIOS 인터럽트 17H번에 연결시켜서 인터럽트 형태로 하드카피를 지원하게 하였다. AH 레지스터에 88H를 넣고, AL레지스터에 옵션을 넣어 준다음 인터럽트 17H를 호출하면 해당하는 모드로 화면을 하드카피해 주게 된다.

AL 레지스터에 넣어 줄 옵션은 DOS상에서 지정해준 옵션과 같다. 단 반전 화면을 찍을 때는 각 옵션에 4를 더해준다. 그러므로 옵션값은 다음과 같게 된다.

① AL=0 : 720✻348 모드 0페이지, 정상 출력, 가로 찍기 (작은 크기)
② AL=1 : 640✻400 모드 0페이지, 정상 출력, 가로 찍기 (작은 크기)
③ AL=2 : 720✻348 모드 1페이지, 정상 출력, 가로 찍기 (작은 크기)
④ AL=3 : 640✻400 모드 1페이지, 정상 출력, 가로 찍기 (작은 크기)
⑤ AL=4 : 720✻348 모드 0페이지, 반전 출력, 가로 찍기 (작은 크기)
⑥ AL=5 : 640✻400 모드 0페이지, 반전 출력, 가로 찍기 (작은 크기)
⑦ AL=6 : 720✻348 모드 1페이지, 반전 출력, 가로 찍기 (작은 크기)
⑧ AL=7 : 640✻400 모드 1페이지, 반전 출력, 가로 찍기 (작은 크기)
⑨ AL=8 : 720✻348 모드 0페이지, 정상 출력, 세로 찍기 (큰 크기)
⑩ AL=9 : 640✻400 모드 0페이지, 정상 출력, 세로 찍기 (큰 크기)
⑪ AL=A : 720✻348 모드 1페이지, 정상 출력, 세로 찍기 (큰 크기)
⑫ AL=B : 640✻400 모드 1페이지, 정상 출력, 세로 찍기

(큰 크기)
⑬AL=C : 720✱348 모드 0페이지, 반전 출력, 세로 찍기
(큰 크기)
⑭AL=D : 640✱400 모드 0페이지, 반전 출력, 세로 찍기
(큰 크기)
⑮AL=E : 720✱348 모드 1페이지, 반전 출력, 세로 찍기
(큰 크기)
⑯AL=F : 640✱400 모드 1페이지, 반전 출력, 세로 찍기
(큰 크기)

어셈블러를 모르는 독자들은 앞의 이야기를 완전히 이해하기 힘들 것이다. 그렇더라도 각 언어별로 사용법을 이야기할테니 앞의 내용은 이해하지 못해도 무방할 것이다.

그러면 이제 언어별로 하드카피 루틴을 사용하는 방법을 설명하겠다. 공통되는 원리는 위의 옵션을 지정해주고 소프트웨어 인터럽트를 호출하는 것이다.

또 한가지의 사용법이 있는데 그것은 인터럽트를 사용하지 않고 다른 언어상에서 하드카피 프로그램을 직접 실행시키는 것이다. 그러기 위해서는 본 하드카피 프로그램이 램상주가 아닌 직접 수행기능을 가져야 할 것이다. 이 때 사용하는 것이, 옵션 'R'이다. 이 옵션을 뒤에 붙여 주면 프로그램은 직접 하드카피를 수행 한후 메모리에 남지 않고 작업을 종료한다. 이 기능을 사용하면 무척 간단하지만 디스크가 잠시 액세스 되게 된다.

이들 옵션은 혼용될 수 있다. 본 프로그램의 옵션 사용의 완전한 형태를 보면 다음과 같다.

```
C> HARCOPY [/] [OPTION] [/] [R]
```

여기서 [ ]로 싸인 부분은 생략 가능한 것이다. 그러므로 720✱348 모드 0페이지를 사용하는 경우는

```
C>HARDCOPY
C>HARDCOPY/0
C>HARDCOPY /0
C>HARDCOPY / 0
C>HARCOPY 0
```

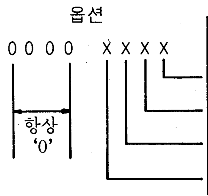
이 모두 같은 기능이 된다. 또한 직접 수행을 시키는 경우는

```
C>HARDCOPY/R
C>HARDCOPY R
C>HARDCOPY/0/R
C>HARDCOPY/0R
C>HARDCOPY /0R
C>HARDCOPY /0 /R
```

등이 모두 같은 기능이다. 단 옵션이 'R'뒤로는 올 수 없다. 그리고 'R'은 'r'로 해도 무방하다.

옵션은 앞서 말한 바와 같이 16진수로 생각하면 된다. 그러므로 이들 옵션 값을 비트맵으로 나타내 보면 그림 1과 같이 된다.

〔그림 1〕 비트맵으로 본 옵션값



기 능	X=0	X=1
모드	720✱348	640✱400
페이지	0	1
인쇄	정상	반전
크기	작게	크게

• 어셈블리어에서의 사용법

어셈블리어에서는 아주 간단하게 호출가능하다.

```
MOV AH,88H
MOV AL,00H   ; 00H 대신 원하는 옵션을 넣어 준다.
INT 17H
```

이러한 간단한 루틴으로 현재의 화면을 하드카피 가능하다. 하드카피를 원하는 부분에 이 루틴을 삽입하면 된다.

• 파스칼에서의 사용

고급언어에서는 크게 두가지 방식을 사용할 수 있다. 한가지는 외부 프로세서(차일드 프로세서)를 호출하는 방식이고 또 한가지는 소프트웨어 인터럽트를 호출하는 방식이다. 전자는 다음과 같이 사용한다.

〈예1〉
```
Uses Dos;
begin
{ 다른 루틴 }
Exec( "hardcopy.com", "/0/r" );
end.
```

여기서 뒤에 붙는 '/0/r'의 '0'는 앞서 언급한 옵션과 같다.

그러므로 0에서 f까지를 사용할 수 있다. 다음에 나오는 '/r'은 (대·소문자를 구분하지 않는다.) 직접 실행 옵션이다. 이 옵션이 붙으면 프로그램은 하드카피 루틴을 램상주시키지 않고 수행한 후 종료한다. 이 방법은 무척 사용이 간단하고 램상주가 되지 않으므로 다른 프로그램과 충돌 위험이 전혀 없을 뿐 아니라 사용자 메모리도 그만큼 더 쓸 수 있게 된다. 하지만 단점은 호출될 때마다 디스크 드라이브가 액세스된다는 점이다. 그러나 매우 짧은 프로그램이므로 별문제가 되지 않을 것이다.

후자의 소프트웨어 인터럽트를 호출하는 방식은 다음과 같이 사용된다.

〈예2〉
```
Uses Dos;
var
  regs : registers;
begin
    { 다른 루틴 }
    regs.ah := $88;
```

```
    regs.al := $04; { 옵션을 4로 하였으므로 720*348 모드 0페이지를
                      반전하여 작게 인쇄 하게 된다.}
    intr( $17, regs );
end.
```

이 방법은 파스칼 프로그램이 수행되기 전에 하드카피 프로그램을 실행시켜 주어야 한다.(아무 옵션이나 관계 없음)

이들중 편리한 방법을 사용하면 된다.

• C에서의 사용

C에서도 다양한 방법으로 호출하여 사용할 수 있다. 우선 exec를 사용하는 방법은 다음과 같다.

```
〈예 1〉 #include <process.h>
        main()
        {
            /* 다른 루틴들 */
            execl("hardcopy","hardcopy","/3/r","0");
        }
```

이 방법은 가장 간단한 방법으로 수행될 때마다 잠시 디스크가 액세스되지만 메모리에 남지 않으므로 편리하다. 여기서 '/3/r'은 '/3R', '/3r', '3R', '3/r' 등으로도 사용가능하다. 옵션을 3으로 하였으므로 640✕400 모드 1페이지를 작게 인쇄하여 줄 것이다.

```
〈예 2〉 #include <dos.h>
       main()
       {
        /* 다른 루틴들 */
          _AH=0x88;
          _AL=0x01; /* 여기에 원하는 옵션을 넣는다. 여기서는 옵션을 1로 하였
                       으므로 640*400 모드 0페이지를 작게 찍어 준다. */
          geninterrupt(0x17);
       }
```

이와 같이 하면 간단하게 하드카피 인터럽트를 호출할 수 있다. 이 방법은 이 루틴이 수행되기 전에 DOS상에서 하드카피 프로그램이 이미 수행되어 있어야 하며 인터럽트를 사용하므로 디스크가 다시 액세스 되지 않아서 좋다. 또 다음과 같은 방법이 있다.

```
〈예 3〉#include <dos.h>
      union REGS regs;
      main()
      {
    resg.h.ah=0x88;
    regs.h.al=0x0f; /* 여기에 옵션을 정해준다. 이상태는 옵션이 F이므로
                       640*400 모드, 1페이지를 반전하여 크게 찍어 준다.*/
    int86(0x17,&regs,&regs);
      }
```

이 방법도 이미 하드카피 프로그램이 수행되어 있어야 한다. 역시 인터럽트를 사용하므로 디스크로부터 다시 읽지 않는다.

앞의 3가지 예가 아니라도 여러가지 방법이 있다. 사용 해보고 자신이 편리한 방법을 선택하기 바란다.

• 베이직에서의 사용

근래의 발전된 베이직들에서는 인터럽트 호출이 가능하지만 다른 베이직과 차이가 있으므로 공통된 것만을 언급하도록 하겠다.

베이직에는 Shell 명령이 있어서 DOS의 외부 프로그램을 호출할 수 있다. 그러므로 본 프로그램의 직접 수행 옵션을 사용하면 간단히 하드카피루틴을 호출할 수 있다. 명령은 다음과 같다.

```
Shell "hardcopy /0/r"
```

여기서 뒤의 옵션은 앞에서 설명한 다른 언어에서의 역할과 같다.

• 기타

근래의 언어들은 대부분 고급 언어상에서 소프트웨어 인터럽트를 발생시킬 수 있도록 되어 있다. 그러므로 매뉴얼을 참조하여 인터럽트 17H를 앞서 말한 옵션으로 발생시키는 방법을 찾아 보기 바란다. 또한 타 프로그램의 호출도 대부분 가능하므로 자신이 사용하는 언어의 기능을 사용하면 될 것이다.

그리고 소스를 컴파일하여 만들어진 오브젝트 화일로 라이브러리를 만들어 두고 링크를 하여 사용하면 더욱 편리할 것이다.

프로그램 설명

리스트 1의 하드카피 프로그램은 어셈블리어로 쓰여 졌다. 처음에는 각 언어로 하드카피 루틴을 만들어 소개할까 하였으나, 사용하기가 번거롭고, 베이직과 같은 경우 컴파일러형 베이직으로 짠 경우도 무척 속도가 느렸다. C나 파스칼에서는 속도로 인한 문제는 없었으나 역시 각 언어별로 작성한다는 것은 경제적이지 못하다는 생각이 들었다.

따라서 본고에서는 어셈블리어로 된 하드카피 프로그램을 메모리에 상주시키고 이를 인터럽트를 사용하여 호출할 수 있게 만들어 다른 프로그램들에서 자유롭게 불러쓸 수 있도록 하였다.

기능을 다시 정리하자면, 화면 모드에 따라서 프로그램을 바꿀 필요가 없도록 가장 많이 사용하는 모드인 720✕348, 640✕400 모드와 0, 1페이지를 모두 지원하고 크기도 가로로 작게 찍어주는 것과 세로로 크게 찍어주는 두가지를 지원한다. 필자는 기존의 몇가지 하드카피 프로그램을 사용해 보았지만 대부분이 화면의 모드를 자동으로 검사하여 그래픽 모드인 경우만 찍어 주도록 하고 있었다.

이러한 것은 편리한 기능이지만 독특한 기교를 사용할 수 없게 된다. 또한 이러한 경우는 사용한 그래픽 프로그램이 착실하게 그래픽 모드를 세트하여 주어야만 된다. 만일 CRTC를 직접 제어하는 방식으로 그래픽을 구현하는 프로그램에서는 전혀 그래픽을 하드카피할 수 없게 된다.

경우에 따라서는 화면에 텍스트 모드를 보여 주면서 그래픽 화면을 찍을 필요도 있을 것이고, 화면에 나타나지 않은 페이지를 찍을 경우도 있을 것이다. 그러므로 모든 것을 사용자가 제어할 수 있도록 하였다. 또한 앞에서도 언급했듯이 화면의 흰점은 실제로 프린터상에서는 검은 점으로 찍히므로 화면을 반전해서 찍을 수도 있도록 하고 있다.

이것은 램상주 형태로 기존의 하드카피 프로그램과 같이 핫키를 사용하여 찍을 수도 있고 다른 프로그램에서 호출하여 사용할 수도 있다.

허클리스에서 자주 사용하는 모드 중에 한가지 독특한 것이 있는데 그것은 CGA 시뮬레이션 모드이다. 이것은 많은 게임들이 CGA를 지원하므로 허클리스에서 그 프로그램들을 사용하기 위해 SIMCGA나 MKFILE 등을 먼저 수행한 경우이다. 이 경우는 본 프로그램에서 지원하지 않는다. 구현이 그리 어려운 일은 아니지만 이미 시중에서 구할 수 있는 하드카피 프로그램들이 많이 있기 때문이며, 자신이 어느 언어로 새로 프로그램을 개발하면서 CGA 시뮬레이션 모드를 사용할 확률이 거의 없기 때문이다.

그러므로 이 모드를 제외한 대부분의 경우 화면에 나타난 그래픽 모드의 해상도와 페이지만 알면 하드 카피를 할 수 있을 것이다. 한글 도깨비를 띄워 놓은 각종 프로그램의 화면이나, NKP, THP 등을 사용한 화면, 닥터할로우, orCAD, 프린터 마스터 등등에서 모두 사용가능하다.

〔그림 2〕 720×348모드 그래픽화면을 하드카피한 것



프로그램의 루틴별 설명은 비교적 자세한 주석을 붙여 놓았으므로 생략하도록 하겠다.

컴파일 방법

리스트 1의 프로그램을 적당한 에디터로 작성하여 MASM이나 TASM로 컴파일한다. 필자는 MASM V5.0을 사용하였다. 그 경우의 컴파일 방법을 화면 1에 나타내었다. 이렇게 하여 생성된 HARDCOPY.COM을 사용하면 된다.

소스를 이용하여 엡슨 호환이 아닌 프린터를 사용하는 경우는 메뉴얼을 참조하여 그 기종에 맞는 그래픽 제어 코드를 사용하면 될 것이다. 또한 생성된 .OBJ 코드를 사용하여 라이브러리를 만들어 사용하거나 링크를 시키면 다른 프로그램에 붙여 사용할 수도 있다.

기타

그림 2, 3, 4에 본 하드카피 프로그램을 사용한 예를 들었다. 그리고 많은 사람들이 사용하는 9핀 프린터 중에 LX-800이 있는데 이 기종은 하드카피를 하면 헤드가 한 줄 이동할 때마다 약간의 빈 줄이 생긴다. 이것은 프린터 자체의 버그 때문이다.

〔화면 1〕 컴파일방법

```
C:\MASM>masm hardcopy;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51554 + 367454 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\MASM>link hardcopy;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

C:\MASM>exe2bin hardcopy.exe hardcopy.com

C:\MASM>dir hardcopy.*

 Volume in drive C has no label
 Directory of  C:\MASM

HARDCOPY BAK    18176   1-01-80    3:10a
HARDCOPY ASM    18304   1-01-80    3:27a
HARDCOPY OBJ     1992   1-01-80    3:27a
HARDCOPY EXE     2265   1-01-80    3:27a
HARDCOPY COM     1497   1-01-80    3:28a
        5 File(s)    385024 bytes free

C:\MASM>
```

〔그림 3〕 그래픽한글 화면을 하드카피한 것

```
   C) 대화의 광장          N) 초보자 안내         U) 설 문 지
   P) 패스워드 변경        X) 매뉴표시/생략       T) 한 화면 멈춤/계속
   W) 현재 사용자          ?) 도 움 말            G) 종 료
   H) 한글코드 선택
==========================================================================

선택: M,F,B,C,N,U,P,X,T,W,G,H or ? ===> f



남은 시간 = 55분 39초

Download 영역 = utility
하루에 download를 받을 수 있는 최대 용량은 262144 bytes

—[ File menu ]————————————————————————————————————————
   A) 영역변경          D) 화일받기             U) 화일보내기
   L) 화일목록          N) 새로운 화일목록      B) Unix형식 목록
   F) 화일찾기          C) 화일내용보기 *       X) 매뉴표시/생략
   Q) 메인메뉴          G) 종료 (전화끊기)      ?) 도움말
——————————————————————————————————————————————————————
* archive된 화일의 내용을 보여줍니다.

선택: A,D,U,L,F,C,X,Q,G, or ? ===>
 ALT-F10  HELP | ANSI-BBS | FDX |  1200 N81 | LOG CLOSED | PRT OFF | CR  | CR
```

〔그림 4〕 640×400모드의 한글화면을 하드카피한 것

```
인터럽트 17h 번에 연결 시켜서 인터럽트 형태로 하드카피를 지원하게 하였다.
AH레지스터에 88h를 넣고, AL레지스터에 옵션을 넣어 준다음 인터럽트 17h를
호출하면 해당하는 모드로 화면을 하드카피 해 주게 된다.
AL 레지스테에 넣어 줄 옵션은 도스상에서 지정해준 옵션과 같다.
단 반전 화면을 찍을 때는 각 옵션에 4를 더해준다. 그러므로 옵션값은 다음과
같게 된다.

  AL = 0  : 720*348 모드 0 페이지, 정상 출력, 가로 찍기 (작은 크기)
  AL = 1  : 640*400 모드 0 페이지, 정상 출력, 가로 찍기 (작은 크기)
  AL = 2  : 720*348 모드 1 페이지, 정상 출력, 가로 찍기 (작은 크기)
  AL = 3  : 640*400 모드 1 페이지, 정상 출력, 가로 찍기 (작은 크기)
  AL = 4  : 720*348 모드 0 페이지, 반전 출력, 가로 찍기 (작은 크기)
  AL = 5  : 640*400 모드 0 페이지, 반전 출력, 가로 찍기 (작은 크기)
  AL = 6  : 720*348 모드 1 페이지, 반전 출력, 가로 찍기 (작은 크기)
  AL = 7  : 640*400 모드 1 페이지, 반전 출력, 가로 찍기 (작은 크기)
  AL = 8  : 720*348 모드 0 페이지, 정상 출력, 세로 찍기 ( 큰  크기)
  AL = 9  : 640*400 모드 0 페이지, 정상 출력, 세로 찍기 ( 큰  크기)
  AL = A  : 720*348 모드 1 페이지, 정상 출력, 세로 찍기 ( 큰  크기)
  AL = B  : 640*400 모드 1 페이지, 정상 출력, 세로 찍기 ( 큰  크기)
  AL = C  : 720*348 모드 0 페이지, 반전 출력, 세로 찍기 ( 큰  크기)
  AL = D  : 640*400 모드 0 페이지, 반전 출력, 세로 찍기 ( 큰  크기)
COMMAND:
C:A.A                 LINE:116    COLUMN:11     INSERT     MODE:TEXT
1HELP  2EDIT  3SAVE  4QUIT  5TEXT  6FORM  7DRAW  8SWITCH 9INSLINE 10PRINT
```

〔리스트 1〕 허클리스 그래픽카드용 하드카피 프로그램

```
;-------------------------------------------------------------------;
;                                                                   ;
;              허클리스 그래픽카드용 하드카피 프로그램              ;
;                                                                   ;
;                 작성자: 김원준,  작성일: 1989/7/4                 ;
;                                                                   ;
;                  소속: 건국 대학교 전자공학과 '87                 ;
;       건국 컴퓨터 연구회[KCA], 아마추어 무선 연구회[HLOAIR]       ;
;           열팔 게시판 ID: dolputer, H-MAIL ID: KCA                ;
;                                                                   ;
;       본 프로그램의 일부 또는 전체를 무단 전재, 복제하는 것은     ;
;            저작권 보호법에 의해 적극 권장되고 있습니다.           ;
;                                                                   ;
;-------------------------------------------------------------------;

Ctrl        equ     04h             ;CTRL키 플래그
Alt         equ     08h             ;ALT키 플래그
Tab         equ     09h             ;Tab 의 아스키코드
Space       equ     20h             ;스페이스의 아스키코드
Slash       equ     2fh             ;'/'의 아스키코드
PadMinus    equ     4ah             ;'-'키의 스캔코드
PadPlus     equ     4eh             ;'+'키의 스캔코드
PrtString   equ     09h             ;DOS 인터럽트 펑션 9번
BiosInt     equ     17h             ;프린터관계 BIOS 인터럽트
Terminate   equ     20h             ;프로그램 수행 종료 인터럽트
DosInt      equ     21h             ;DOS 펑션 콜
Resident    equ     27h             ;램상주 인터럽트
IntCon      equ     20h             ;인터럽트 콘트롤러(8259)번지
PortA       equ     60h             ;PPI 8255의 포트 A
PortB       equ     61h             ;PPI 8255의 포트 B
Option      equ     80h             ;옵션 베이스
KeyFlag     equ     417h            ;키 상태 플래그 번지
SimModeX    equ     50h             ;640*400 모드용 데이터
SimModeY    equ     32h             ;   (작게 인쇄)
```

1

```
StdModeX    equ     5ah             ;720*348 모드용 데이터
StdModeY    equ     2ch             ;   (작게 인쇄)
StdColY     equ     5ah             ;640*400 모드용 데이터
StdColX     equ     15ch            ;   (크게 인쇄)
SimColY     equ     50h             ;720*348 모드용 데이터
SimColX     equ     190h            ;   (크게 인쇄)
HighBit     equ     80h             ;인터럽트 리셋트 플래그
Invert      equ     0ffh
BaseOffSet  equ     2000h           ;그래픽 어드레스 계산용
PageLength  equ     4000h           ;그래픽 페이지의 워드수
SetPage0    equ     0b000h          ;그래픽 페이지0의 세그먼트

;-------- 레지스터 보관 매크로 -------------------------------------------;

PushAll     macro                   ;모든 레지스터와
            push    ds              ;플래그 저장
            push    es
            push    di
            push    si
            push    ax
            push    bx
            push    cx
            push    dx
            pushf
            endm

;-------- 레지스터 복구 매크로 -------------------------------------------;

PopAll      macro                   ;모든 레지스터와
            popf                    ;플래그 복구
            pop     dx
            pop     cx
            pop     bx
            pop     ax
```

2

3

```
        pop     si
        pop     di
        pop     es
        pop     ds
        endm

;-------- 초기 스타드 ------------------------------------------------------;

CodeSeg segment
        assume  cs:CodeSeg,ds:CodeSeg

        org     100h

Start:  jmp     Install                 ;램상주 루틴으로 점프

;-------- Work Area -------------------------------------------------------;

OldInt9     dd  ?                       ;원래의 INT 9H 값이 보관된다.
OldInt17    dd  ?                       ;원래의 INT 17H 값이 보관.
DosOption   db  ?                       ;각 옵션 값이 저장된다.
PosX        db  ?                       ;가로 좌표 (작게 인쇄)
PosY        db  ?                       ;세로 좌표
Buffer      db  8 dup(?)                ;8비트 블록이 저장된다.
SetUpData   db  0ah,1bh,'A',08h,1bh,'L',0d0h,02h,'$'
            db  0ah,1bh,'A',08h,1bh,'L',080h,02h,'$'
LargeData   db  0aH,1bH,'A',08H,1bh,'L',0b8H,02H,'$'
            db  0aH,1bH,'A',08H,1bh,'L',020h,03h,'$'
Mesg01:     db  ' <<< HGC Screen HardCopy Utility '
            db  '--- Option Usage >>> ',10,13,7,10
            db  ' C>HARDCOPY [/] [OPTION] [/] [R] '
            db  '-- Direct Execution',10,13,9,9,'    ¦',10,13
            db  9,9,'7   ¦   0  +------+---------+---------+'
            db  10,13,'      <Option>  0000 XXXX  '
            db  '¦ Func ¦   X=0   ¦   X=1   ¦',10,13
            db  '       Hex: 0-F  ¦  ¦ ¦¦¦¦  '
```

4

```
            db  '+------+---------+---------+',10,13
            db  9,9,'+--+ ¦¦¦+--¦ Mode ¦ 720*348 ¦ 640*400 ¦',10,13
            db  9,9,'¦     ¦¦+---¦ Page ¦    0    ¦    1    ¦',10,13
            db  9,9,'Only ¦+----¦ B/W  ¦ Normal  ¦ Inverse ¦',10,13
            db  9,9,'Zero +-----¦ Size ¦  Small  ¦  Large  ¦',10,13
            db  9,9,9,'   +------+---------+---------+',10,10,13
            db  '* Keys: [Ctrl]+[+]: Normal, [Ctrl]+[-]: Inverse'
            db  10,13,'* Bios Inturrupt support:'
            db  'Int 17h, ah:88h, al:Option 0h - 0fh',10,13
            db  '* By Kim Weon Jun [1989/7]',10,13,'$'
Mesg02      db  'HardCopy utility already installed '
            db  '& parameter updated',10,13,'$'
Mesg03      db  'HardCopy utility installed. '
            db  '[Help: C>HARDCOPY /?]',10,13,'$'

;-------- 새로운 인터럽트 17H ---------------------------------------------;

NewBios     proc    near

            cmp     ah,88h              ;하드카피를 위한 호출인가?
            jz      NewProcess          ;하드카피 루틴 호출
            jmp     cs:OldInt17         ;원래의 인터럽트로

NewProcess: cmp     al,88h              ;옵션 변경을 위한 호출인가?
            jz      UpDate              ;옵션 변경 루틴으로
            cmp     al,0                ;옵션 범위 조회
            jb      IntQuit             ;범위를 벗어난 경우
            cmp     al,0fh
            jg      IntQuit

            PushAll                     ;레지스터 보관

            push    cs                  ;데이터 세그먼트 정의
            pop     ds
```

```
        mov     DosOption,al            ;옵션 저장
        cmp     al,7                    ;하드카피의 크기?
        jg      Large                   ;크게 인쇄 (돌려 찍기)
        call    HardCopy                ;작게 인쇄 (가로 찍기)
        jmp     Ship                    ;종료

Large:  call    LargeCopy
        jmp     Ship

UpDate: mov     cs:DosOption,dl         ;옵션 변경

IntQuit: xor    al,al
        iret                            ;인터럽트 종료

NewBios endp

;-------- 메인 루틴 -----------------------------------------------------;

Main    proc    near

        PushAll                         ;레지스터, 플래그 저장

        sti                             ;하드웨어 인터럽트 허가
        cld                             ;리피트 프리픽스 증가방향 설정

        push    cs                      ;DS정의
        pop     ds

        xor     ax,ax                   ;ES를 ROM BIOS 세그먼트로 설정
        mov     es,ax

        test    byte ptr es:KeyFlag,Ctrl ;콘트롤 키가 눌려 졌는가?
        jz      Skip                    ;아니면 원래의 INT 9H루틴으로

        test    byte prt es:KeyFlag,Alt ;ALT키가 눌려 있지 않은가?
```

5

```
        jnz     Skip

        in      al,PortA                ;8255로부터 키 스캔코드를

        cmp     al,PadPlus              ;키패드의 '-'가 눌려졌는가?
        jz      Process                 ;맞으면 하드카피 루틴으로

        cmp     al,PadMinus             ;키패드의 '+'가 눌려졌는가?
        jnz     Skip                    ;아니면 원래의 INT 9H루틴으로

        or      DosOption,04h           ;반전 플래그 세트
Process: test   DosOption,08h           ;하드카피의 크기?
        jnz     Great                   ;크게 인쇄

        call    HardCopy                ;작게 인쇄
        jmp     EndCopy

Great:  call    LargeCopy               ;크게 인쇄

EndCopy: and    DosOption,0fbh          ;반전 플래그 클리어

;-------- 하드웨어 인터럽트 클리어 -----------------------------------------;

        in      al,PortB                ;인터럽트 콘트롤러에
        mov     ah,al                   ;처리 종료를 알린다.
        or      al,HighBit
        out     PortB,al
        mov     al,ah
        out     PortB,al
        mov     al,20h
        out     IntCon,al
Ship:   PopAll                          ;레지스터, 플래그 복구
        iret                            ;인터럽트 종료
```

6

7

```
;-------- 원래의 키보드 인터럽트 ------------------------------------------;

Skip:       PopAll                      ;플래그, 레지스터 복구
            jmp     cs:OldInt9          ;원래의 인터럽트 루틴으로 점프

Main        endp

;-------- 하드카피 루틴 (작게 인쇄) ----------------------------------------;

HardCopy    proc    near

            mov     bx,SimModeX         ;640*400 모드 설정
            mov     cx,SimModeY

            test    DosOption,1
            jnz     SetEnd

            mov     bl,StdModeX         ;720*348 모드 설정
            mov     cl,StdModeY

SetEnd:     mov     PosY,ch             ;세로 좌표 클리어 (ch=0)

PrtRow:     push    cx
            mov     PosX,ch             ;가로 좌표 클리어

            lea     si,SetUpData        ;프린터 셋업 베이스 어드레스

            call    PrtSetUp            ;프린터를 그래픽 모드로 설정

            mov     cl,bl
PrtLine:    push    cx
            call    Get8bit             ;비디오 메모리로 부터 8*8
                                        ;블록을 읽어 들인다.

;------- 읽어 들인 데이터를 인쇄용으로 변환 --------------------------------;
```

8

```
            mov     cl,8                ;8바이트씩 블록으로 처리한다.
Iloop:      push    cx
            mov     dl,cl

            lea     si,buffer           ;읽어 들인 데이터가 저장된 곳
            xor     ah,ah
            mov     cl,8                ;비트처리를 위함(8비트=1바이트)
Jloop:      push    cx

            lodsb                       ;한바이트를 읽어 온다.
            mov     cl,dl               ;8비트 블록의 데이터를
            ror     al,cl               ;읽어서 각 비트(8*8)의
            and     al,HighBit          ;행과 열을 바꾸어 준다.
            pop     cx
            push    cx
            rol     al,cl
            or      ah,al               ;변환된 한 바이트를 누적

            pop     cx
            loop    Jloop

            mov     al,ah
            call    PrtData             ;한 바이트를 인쇄한다.
            pop     cx
            loop    Iloop

            inc     PosX                ;가로 좌표를 증가
            pop     cx
            loop    PrtLine             ;가로 인쇄를 계속
            inc     PosY                ;세로 좌표 증가
            inc     PosY
            pop     cx
            loop    PrtRow              ;모두 찍을 때 까지 반복

PrtClear:   mov     al,1bh              ;프린터를 그래픽 모드로 부터
```

```
        call    PrtData             ;해제시키고 초기화 한다.
        mov     al,'@'
        call    PrtData
        ret                         ;처리 종료
HardCopy endp

;-------- 현재의 위치에 있는 한블록을 읽음 ---------------------------------;

Get8bit proc    near

        mov     ax,SetPage0         ;그래픽 페이지0을 설정

        test    DosOption,2
        jz      ModePage0
        add     ah,08h              ;페이지 1로 변경

ModePage0: mov  es,ax               ;보조 세그먼트 정의

        lea     si,buffer           ;읽은 데이터를 저장할 곳

        mov     cl,PosX             ;현재의 포인터에 있는
        mov     ax,bx               ;8바이트의 데이터를
        mul     PosY                ;읽어서
        add     ax,cx               ;버퍼에 저장한다.
        mov     di,ax

        mov     cl,2
Bytes:  push    cx
        mov     dl,cl
        mov     cl,4
Nibble: mov     al,es:[di]          ;비디오 메모리를 읽는다.

        mov     dh,DosOption
        test    dh,4                ;반전 모드 인가?
        jz      Normal
```

9

```
        xor     al,Invert           ;데이터를 반전 한다.

Normal: test    dh,1                ;720*348 모드인가?
        jnz     NonEnd

        cmp     PosY,86             ;마지막 열인가?
        jnz     NonEnd

        cmp     dl,1
        jnz     NonEnd
                                    ;348열은 8로 나누어 지지
        xor     al,al               ;않으므로 엉터리 데이터가
                                    ;들어가지 않도록 한다.
NonEnd: mov     [si],al             ;버퍼에 저장한다.
        inc     si
        mov     ax,2000h            ;비디오 메모리를 계산한다.
        add     di,ax
        loop    Nibble
        mov     al,bl
        mov     ah,HighBit
        add     di,ax
        pop     cx
        loop    Bytes
        ret
Get8bit endp

;-------- 하드카피 루틴 (크게 인쇄) ----------------------------------------;

LargeCopy proc  near

        mov     ax,SetPage0         ;그래픽 페이지를 0으로 설정

        test    DosOption,2         ;페이지 1모드 인가?
        jz      PageZero
        add     ah,08h              ;페이지1로 고침
```

10

11

```
PageZero:   mov     es,ax               ;보조 세그먼트 설정

            mov     di,SimColX          ;640*400 모드용 데이터
            mov     bx,SimColY

            test    DosOption,1         ;720*348모드인가?
            jnz     EndSet
            mov     di,StdColX          ;720*348 모드용 데이터
            mov     bl,StdColY

EndSet:     mov     PosY,0              ;세로 포인터 클리어

RowPrt:     lea     si,LargeData        ;프린터 셋업 데이터 베이스
            call    PrtSetUp            ;프린터를 그래픽 모드로 설정
            mov     cx,di
ColPrt:     mov     ax,cx               ;Column MOD 4 -> AX
            dec     ax
            and     ax,0003
            mov     dx,2000h            ;데이터의 번지를 계산
            mul     dx
            add     al,PosY
            push    ax
            mov     ax,cx
            dec     ax
            shr     ax,1
            shr     ax,1
            mul     bx
            pop     dx
            add     ax,dx
            mov     si,ax
            mov     al,es:[si]          ;계산된 번지로 부터 읽음

            test    DosOption,4         ;반전 모드 인가?
            jz      NormalPrt
            xor     al,Invert           ;데이터 반전
```

12

```
NormalPrt:  call    PrtData             ;한줄 인쇄
            call    PrtData             ;한번 더 인쇄
            loop    ColPrt

            inc     PosY                ;열 포인터 증가
            cmp     PosY,bl             ;다 찍었는가?
            jnz     RowPrt              ;계속 인쇄
            jmp     PrtClear            ;종료

LargeCopy   endp

;-------- 프린터 출력 루틴 ------------------------------------------------;

PrtData     proc    near                ;프린터로 AL의 값을 출력
            xor     dx,dx               ;프린터 번호:0
            xor     ah,ah
            int     BiosInt
            ret
PrtData     endp

;--------- 프린터를 그래픽 모드로 설정 ------------------------------------;

PrtSetUp    proc    near

            test    DosOption,01h       ;그래픽 모드 확인
            jz      PrtLoop
            add     si,9
PrtLoop:    lodsb                       ;셋업 데이터를 읽는다.
            cmp     al,'$'              ;데이터의 끝인가?
            jz      RetSet
            call    PrtData
            jmp     PrtLoop
RetSet:     ret

PrtSetUp    endp
```

```
;-------- 램상주 루틴 -------------------------------------------------------;

Install     proc    near

            cld                             ;리피트 방향 설정
            push    cs                      ;DS 정의
            pop     ds
            mov     si,Option               ;옵션 베이스

            xor     dx,dx
            lodsb                           ;옵션이 없는가?
            cmp     al,00h
            jz      Correct

FetchOne:   lodsb                           ;한 바이트 리드
            cmp     al,Space                ;스페이스면 다음 문자 리드
            jz      FetchOne
            cmp     al,Tab                  ;탭이면 다음 문자 리드
            jz      FetchOne
            cmp     al,Slash                ;슬래시면 다음 문자 리드
            jz      FetchOne

            cmp     al,'0'                  ;옵션의 범위 검사
            jb      Invailed
            cmp     al,'9'
            jg      Invailed
            sub     al,30h
            jmp     Correct

Invailed:   or      al,20h                  ;대/소문자의 구분을 없앰
            cmp     al,'r'                  ;직접 실행 옵션?
            jz      DirectExec
            cmp     al,'a'                  ;영문자를 16진수로 처리
            jb      Wrong
            cmp     al,'f'
```

13

```
            jg      Wrong
            sub     al,57h
            jmp     Correct

Wrong:      lea     dx,Mesg01               ;옵션에 잘못이 있을 경우
PrtMesg:    mov     ah,PrtString            ;안내문 출력
            int     DosInt
            int     Terminate               ;수행 종료

Correct:    mov     dl,al
            mov     DosOption,al            ;옵션 저장

FetchByte:  lodsb
            cmp     al,Space                ;스페이스면 다음 문자 리드
            jz      FetchByte
            cmp     al,Tab                  ;탭이면 다음 문자 리드
            jz      FetchByte
            cmp     al,Slash                ;슬래시면 다음 문자 리드
            jz      FetchByte

            or      al,20h
            cmp     al,'r'
            jnz     HotKey

DirectExec: PushAll

            test    DosOption,08h
            jnz     Turn

            call    HardCopy                ;작게 인쇄
            jmp     Ending

Turn:       call    LargeCopy

Ending:     PopAll
```

14

```
        ret

HotKey: mov     ax,8888h                ;인스톨 검사
        int     BiosInt                 ;옵션 변경
        cmp     al,88h
        lea     dx,Mesg02               ;메시지 출력
        jnz     PrtMesg

        mov     ax,3509h                ;INT 9H의 Segment -> ES
        int     DosInt                  ;       OffSet  -> BX
        mov     word ptr OldInt9,bx     ;원래의 INT 9H값 보관
        mov     word ptr OldInt9[2],es
        mov     ah,25h                  ;새로운 INT 9H 위치지정
        lea     dx,Main
        int     DosInt

        mov     ax,3517h                ;새로운 인터럽트 17H정의
        int     DosInt
        mov     word ptr OldInt17,bx
        mov     word ptr OldInt17[2],es
        mov     ah,25h
        lea     dx,NewBios
        int     DosInt

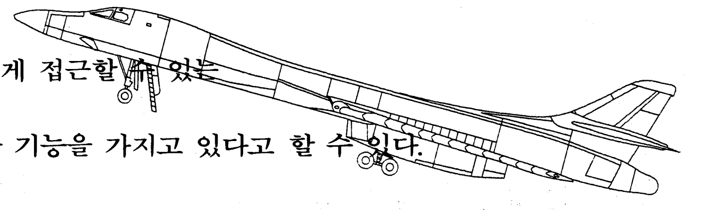
        lea     dx,Mesg03               ;인스톨 메시지 출력
        mov     ah,PrtString
        int     DosInt
        sti                             ;인터럽트 가능
        lea     dx,Install              ;램상주
        int     Resident
Install endp

CodeSeg ends
        end     Start                   ;끝
;------------------------------------------------------------------------; 15
```



누구나 쉽게 사용할 수 있는 CAD 프로그램
프로디자인 Ⅱ

심성호 / 명지대학교 P.R.D.A



CAD는 고급 전문가들만의 전유물인가?
이런 욕구불만을 갖고 있는 일반 PC사용자들이 좀더 손쉽게 접근할 수 있는 프로그램이 프로디자인이다.
한마디로 프로그램 구성은 간단히 되어 있으면서도 강력한 기능을 가지고 있다고 할 수 있다.

프로디자인의 신상명세서

사람들은 개개인마다 선입견이라는 것을 가지고 있다. 저 사람은 마음이 좋을 것 같다든가 저 사람은 간첩같이 생겼다든지 ..등 모든 이에게는 그 모습에서 풍기는 이미지가 있기 마련이다. 대부분의 해커는 그의 모습이 약간은 졸리운 모습이다. 왜냐하면 누군가가 역사는 밤에 이루어 진다고 했듯이 대부분의 해커는 밤에 활동을 하고 낮에는 잠을 자는 올빼미 생활을 반복하기 때문이다.

지금하는 이야기는 해커 이야기가 아니고 프로그램 역시 선입견을 가지고 있다는 것이다. 대단히 좋아 보이는(특히 로고화면) 프로그램이 실은 그렇게 좋지 않은 경우도 많이 보아왔고 겉은 그럭저럭 돌아갈 것 같은 프로그램이 그 기능면에서는 우리를 놀라게 하는 프로그램 또한 많다. 프로디자인(PRODESIGN Ⅱ)를 처음 보았을 때 역시 외면당할만 하다는 생각이 들었다. 그도 그럴 것이 화면에는 그저 네모난 박스안에 조그만 커서 하나만 깜빡이고 있으니 무슨 프로그램인지, 그리고 어떻게 사용하는지 알 길이 없었다. 그런 후 며칠을 사용해 본 결과 그것이 정말 훌륭한 프로그램이라는 것과 이 안에 들어 있는 많은 기능은 캐드라고 이름짓기에 결코 손색이 없음을 알게 되었다.

캐드의 대명사라면 autoCAD가 있다. 그러나 이것을 사용하기에는 무척이나 많은 제약이 따른다. 우선 전문성을 띠고 있기 때문에 사용되는 기종이 한정되어 있으며 최소한 IBM-PC / AT급이어야 하고 하드디스크 없이는 생각조차 하기 어렵다. 또 사용법 또한 힘들어 일반 서민(?)이 쉽게 접할 수 있는 소프트웨어라고 보기 힘들다.

이에 비하여 프로디자인은 디스켓 3장으로 구성되어 있어(실질적으로는 2장) 일반 플로피디스크 드라이브에서도 충분히 사용할 수 있다는 장점이 있다. 또 기능도 일반 캐드에 뒤지지 않아 일반 사용자가 원하는 것은 무엇이든지 만들어낼 수 있다. 서론이 너무 긴 것 같다. 그러면 이렇게 침이 마르도록 칭찬을 하는 프로디자인의 신상명세서를 보도록 하자.

프로디자인은 1986년 미국의 스몰 비지니스 컴퓨터스 (Small Business Computers)라는 회사에서 태어났다. 그러고 보면 나이가 얼마 안되는 셈이다. 그 특징으로는 다음과 같이 몇가지로 나누어 볼 수 있다.

(1) 주변장치의 다양성

프로디자인은 다른 친구들과도 잘 어울린다. 모든 캐드 프로그램이 그렇듯이 프린터, 플로터는 물론 디바이스 포인터로 마우스, 디지타이저 등을 지원한다. 또 모든 그래픽 카드를 지원하여 어떤 기종의 컴퓨터라도 사용을 가능하게 지원하고 있다.

(2) 프린트 기능

프로디자인은 또 멋을 안다. 프린트 방법은 다른 캐드 프로그램과는 다른 방식을 택하고 있다. 필자가 이 프로디자인을 사용하여 프린트한 결과 놀라지 않을 수 없었다. 심지어는 샘플로 작성한 것을 프린팅하여 친구들에게 플로터로 찍은 것이라고 했더니 모두가 믿어버릴 정도이니 말이다.

프로디자인은 프린트할 도면의 밀도를 결정할 수 있는데 9핀 프린터에서는 NORMAL, HIGH, VERY HIGH의 3가지 모드를 지원하고 있다. VERY HIGH의 경우 프린터 헤드가 3번을 움직이게 되므로 자세히 보아도 도트하나 보이지 않는 것은 당연한 일이다. (24핀 프린터에서는 이 모드가 지원되지 않음)

(3) 간편한 도움말 화면

비록 소프트웨어 자체가 메뉴방식으로 되어 있지는 않지만 모든 기능에 대한 도움말이 자세히 나와 있다. 단 한가지 단점이라면 메뉴방식이 아니기 때문에 초보자의 경우, 처음에 어떻게 하여야 도움말 화면을 얻을 수 있는가

하는 것이 문제이다. 현재 우리나라에도 지적 소유권이 적용되고 있어 이 프로그램의 매뉴얼을 구하기는 어려운 것으로 안다. 도움말을 보려면 보통 상태에서 '?'키를 누르면 된다.

(4) UNDO 기능

시트 작성시 〈ESC〉키만 누르면 제일 마지막으로 작성하였던 부분만을 삭제할 수 있다. 이런 기능이 없다면 프로그램을 이용하는데 무척 어려웠을 것이다. 또한 임의의 부분(영역)을 지울 수 있는 지우기 명령도 구비하고 있다.

(5) 눈금자 표시 기능

화면에 원하는 그림을 구상하고 구도를 잡으려면 눈금자가 같이 있어야 함은 지극히 당연하다. 프로디자인에서는 모두 3가지의 표현방식이 있는데 먼저 격자방식이 있고 주변만 눈금 표시하는 방식, 그리고 도트로 나타내는 방식이 있다. 그러므로 그때그때 원하는 모양을 택하여 사용할 수 있다.

(6) 커서 표시 기능

커서의 표시방법도 2가지가 있다. 하나는 일정한 크기의 커서를 사용하는 것이며 다른 하나는 십자의 형태를 사용하는 방법이다. 커서의 모양을 바꾸려면 오른쪽 숫자 패드의 '+'키를 사용하면 된다.

(7) 자유로운 텍스트 표현

나중에 다시 설명을 하겠지만 사용자가 원하는 만큼의 글씨를 원하는 위치에 원하는 모양으로 구성할 수 있다. 예를 들어 세로로 쓰는 글씨는 물론 오른쪽의 높이와 왼쪽의 높이가 다른 글씨도 자유자재로 표현할 수 있다. 또 사용자가 임의로 기울기를 지정할 수 있어 누운 글씨도 사용할 수 있다.

(8) 화일 포맷의 변경

프로디자인은 결코 혼자서 놀지는 않는다. 프로그램안에는 유틸리티가 몇 개 있는데 autoCAD에서 사용하는 데이터 형식으로 변형이 가능하다.

(9) 화면의 회전기능

프로디자인은 이미지를 사용자가 원하는 각도나 원하는 크기로 자유롭게 만들어 준다.

이상으로 프로디자인의 신상 명세서를 알아 보았다. 이제 좀 더 자세히 살펴보기로 하자.

디스켓의 구성

프로디자인(Ⅱ) 디스켓은 3장으로 구성되어 있다. 그러나 하나는 8087 보조프로세서를 사용하는 시스템을 위해 별도로 만든 프로그램이므로 사실상은 2장의 디스켓으로 구성되어 있는 셈이다. 일반 상가에서 구할 수 있는 디스켓은 1번과 2번 디스켓인데 이것만으로도 사용이 가능하다.

디스켓 안에 들어 있는 화일들을 살펴보면 다음과 같다.

• 디스켓 #1

PD.EXE : 프로디자인의 메인 프로그램

PDPRINT.EXE : 프로디자인의 프린팅 프로그램

PDSETUP.EXE : 프로디자인을 기종에 맞게 셋업해 주는 프로그램

PDESIGN1.VCT : 텍스트 문자를 표현하는 텍스트 폰트 화일

PDESIGN2.VCT : 텍스트 문자를 표현하는 텍스트 폰트 화일

PDESIGN.SYS : 프로디자인을 실행할 때 사용되는 셋업 화일

• 디스켓 #2

MOUSE.COM : 마우스 드라이버 프로그램(마우스 사용시)

HPGL.EXE : HP플로터를 사용하도록 데이터를 변환해주는 유틸리티

IGES.EXE : 규모가 큰 캐드시스템에서 사용할 수 있도록 포맷을 변경해 주는 유틸리티

*.PDI : 프로디자인의 샘플 시트

프로그램 실행

프로디자인 프로그램을 실행하기 전에는 실행할 시스템 환경에 맞게 셋업을 시켜 주어야 한다. 여기서는 주로 프린터 환경과 포인팅 디바이스, 그리고 비디오 보드 등을 선택하게 된다.

프로디자인 디스켓 #1을 A드라이브에 넣고 다음과 같이 입력한다. (하드디스크를 사용하는 경우는 적당한 디렉토리를 만들어 디스켓 2장의 내용을 모두 복사한 후 그 디렉토리에서 이 셋업 프로그램을 실행시키면 된다.)

PDSETUP ↵

(↵표시는 리턴키를 누르라는 뜻임)

PDSETUP프로그램을 실행시키면 화면 1과 같이 프린터 기종과 포인팅 디바이스 및 비디오 보드 등을 묻는 메시지가 나온다. 선택 지정이 모두 끝나면 새로 지정된 환경을 디스크에 저장하고 DOS 프롬프트로 다시 빠져 나오게 된다. 특히 포인팅 디바이스로써 마우스를 선택하면 시트작업을 할 때 마우스와 키보드를 모두 사용할 수 있어 무척 편리하다.

셋업이 끝나면 프로디자인의 메인 프로그램을 실행시킨다.

PD ↵

앞서 잠시 언급했듯이 프로디자인은 메뉴방식이 아니기 때문에 프로그램을 실행시키면 처음부터 직접 워크시트 화면이 나오게 되어 화면 상단의 상태라인과 십자 모양의 커서만이 보이게 된다.

이 상태에서 그림 그리기는 시작되며 도움말이 필요하면 키보드의 '?' 키를 누르면 된다.

주요 명령어

프로디자인의 명령어는 표1에 요약했으므로 참고로 하기 바라며 여기서는 일반적으로 많이 사용하는 기능만

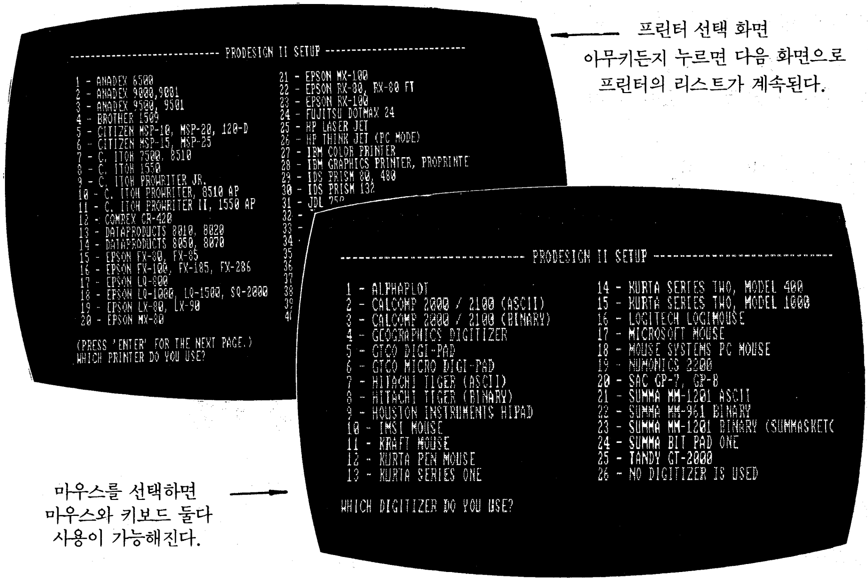
간략히 설명하겠다. 다시 말하지만 프로디자인은 사용하기 쉽도록 설계되어 있어서 다음에 보이는 몇가지 예제를 해보면, 명령어 일람표만 있으면 혼자서도 각종 그래픽을 자유자재로 설계할 수 있을 것이다.

- **커서 바꾸기 : 키보드의 우측에 있는 '+'키를 이용한다.**
- **눈금자 표시 바꾸기 : 'Q'키를 누르면 여러 상태표시가 나오는데 그중에 GRID TYPE을 정의하면 된다.**

0–무표시
1–격자표시
2–주변만 눈금표시
3–점으로 표시

- **커서 이동 : 일반적으로 화살표 키를 사용하나 다음과 같은 경우도 있다.**

〔화면 1〕 셋업 프로그램 (PDSETUP. EXE) 의 실행



프린터 선택 화면
아무키든지 누르면 다음 화면으로
프린터의 리스트가 계속된다.

마우스를 선택하면
마우스와 키보드 둘다
사용이 가능해진다.

〔표 1〕 프로디자인 명령어 요약

(주) 여기서 (R)은 필수로 하여야 할 점을 말하며 (O)는 선택적으로 하는 것을 말한다.

명령	기능	사용점
A : ARC COMMAND	세점에 의한 호를 그린다.	점1(R) : 호의 중심점 점2(R) : 호의 시작점 점3(R) : 호의 끝점
B : BACKWARD ARC	세점에 의한 반대편의 호를 그린다.	점1(R) : 호의 중심점 점2(R) : 호의 시작점 점3(R) : 호의 끝점
C : CURVE COMMAND	세점 이상의 점을 연결하는 곡선을 그린다.	
D : SECTION DELETE COMMAND	지정된 블록 내의 모든 부분을 삭제한다.	점1(R) : 삭제할 블록의 코너를 지정한다. 점2(R) : 삭제할 블록의 맞은편 코너를 지정한다.
E : ERASE COMMAND	하나 또는 여러 개의 포인트를 지정하였을 경우 그 지점에 그려진 모든 그림을 삭제한다.	
F : FILLET COMMAND	각진 부분을 지정하여 곡선으로 만들어 준다.	
H : COLOR COMMAND	화면에 나타나는 색상을 변경한다. 포인트는 필요없다.	
J : STATUS LINE FORMAT COMMAND	스크린 상단의 상태 표시형식을 바꾼다. (1~3)	
K : OVERLAY COMMAND		
L : LAYER COMMAND		
M : SECTION MOVE COMMAND	지정된 블록을 다른 곳으로 이동시킨다.	점1(R) : 블록의 코너를 지정한다. 점2(R) : 블록의 맞은편 코너를 지정한다. 점3(R) : 이동시킬 부분의 좌측하단을 지정한다. 점4(O) : 이동시킬 부분의 우측하단을 지정한다. 점5(O) : 이동시킬 부분의 좌측상단을 지정한다. 점6(O) : 이동시킬 부분의 우측상단을 지정한다.
N : SECTION COPY COMMAND	지정된 블록을 다른 곳으로 복사한다.	점1(R) : 블록의 코너를 지정한다. 점2(R) : 블록의 맞은편 코너를 지정한다. 점3(R) : 이동시킬 부분의 좌측하단을 지정한다. 점4(O) : 이동시킬 부분의 우측하단을 지정한다. 점5(O) : 이동시킬 부분의 좌측상단을 지정한다. 점6(O) : 이동시킬 부분의 우측상단을 지정한다.

명 령	기 능	사 용 점
O : OVAL COMMAND	원 또는 타원을 작성한다.	점1(R) : 원 또는 타원의 중심을 지정한다. 점2(R) : 원 또는 타원의 반지름을 지정한다. 점3(R) : 타원에서 또다른 지나는 점을 지정한다.
P : PAINT COMMAND	지정된 부분의 내부를 칠한다.	점1(R) : 칠하여질 부분의 내부이다.
Q : SYSTEM STATUS COMMAND	현재 지정된 여러 상황을 보여주고 결정한다.	
R : ROTATE COMMAND	지정된 부분 또는 화면 전체를 회전시킨다.	
T : TEXT COMMAND	화면에 문자를 표현한다.	점1(R) : 좌측 하단 점을 지정한다. 점2(O) : 우측 하단 점을 지정한다. 점3(O) : 좌측 상단 점을 지정한다. 점4(O) : 우측 상단 점을 지정한다.
U : UNITS COMMAND	거리의 단위처리를 수행한다.	
V : VECTOR COMMAND	하나 또는 여러 지정점을 라인으로 연결한다.	
W : LINE WIDTH COMMAND	라인이나 다른 모든 기능에서 선의 두께를 결정한다.	
X : SECTION SAVE COMMAND	특정 부분을 블록으로 지정하여 디스크에 저장한다.	점1(R) : 블록의 코너를 지정한다. 점2(R) : 블록의 맞은편 코너를 지정한다.
Y : CLEAR SCREEN COMMAND	화면의 모든 점을 삭제한다.	
Z : ZOOM COMMAND	현재의 작성된 형태를 확대 또는 축소하여 보여준다.	
] : BOX COMMAND	지정된 두점을 대각으로 하는 사각형을 그린다.	점1(R) : 사각형의 코너를 지정한다. 점2(R) : 사각형의 맞은편 코너를 지정한다.
> : ARROW COMMAND	두점간에 화살표와 거리를 표시한다.	
) : SEMI-CIRCLE COMMAND	두점을 지나는 반원을 그린다.	
(: QUARTER-CIRCLE COMMAND	두점을 지나는 4분의 1인 원을 그린다.	
@ : AUTO DIMENSION COMMAND	두지점 간의 거리를 표시한다.	
ALT-@ : AUTO DIMENSION COMMAND, FEET INCHES	두지점간의 거리를 표시하는 것은 앞의 것과 같으나 표기방식이 다르다.	
# : ATTRIBUTE COMMAND	그부분의 속성을 나타낸다. 이것은 Q-KEY에 의해서 결정되는데 속성이 YES인 경우에만 나타난다.	
/ : LINE TYPE COMMAND	선의 형태를 지정한다.	
, : GRAVITY CURSOR MOVE COMMAND	현재 있는 곳에서 가장 가까운 포인트로 커서를 이동시킨다.	
. : GRAVITY POINT COMMAND	현재 있는 곳에서 가장 가까운 포인트로 커서를 지정한다.	
: : SET POINT USING COORDINATES	수치의 입력으로 커서를 원하는 위치로 이동시킨다.	
; : SET POINT USING DISTANCE AND ANGLE	극좌표 형식으로 거리와 각도를 입력받아 다음의 좌표를 지정한다.	
* : POINT CHANGE COMMAND	이미 지정된 포인트를 다른 곳으로 수정할 때 사용된다.	
+ : CROSSHAIR COMMAND	커서를 십자가 형태로 만든다.	
? : HELP COMMAND	도움말을 보여준다.	
F1-F4 : SCREEN WINDOW COMMAND	화면을 이동시킨다. 단, 이 기능은 IBM CGA 에서만 가능하다.	
F5 : PRINT COMMAND	작성한 화일을 프린터로 출력한다.	
F6 : PLOT COMMAND	작성한 화일을 플로터로 출력한다.	
F7 : DIRECTORY COMMAND	작성된 화일의 목록을 보여준다.	
F8 : STOP COMMAND	프로그램을 종료시킨다.	
F9 : RETRIEVE COMMAND	작성한 화일을 디스크에 저장한다.	
F10 : STORE COMMAND	디스크에 저장된 화일을 불러온다.	
ESC : UNDO COMMAND	최근에 작성한 부분을 취소시킨다.	
! : OOPS COMMAND	마지막으로 취소한(ESC키로) 부분을 다시 재생	

〈Ctrl〉-〈→〉: 최 우측으로 이동
〈Ctrl〉-〈←〉: 최 좌측으로 이동
- **포인트 지정**: 포인트 지정은 〈INS〉키를 사용한다. 만약 포인트가 잘못되었을 경우는 〈ESC〉키로 한 포인트씩 취소한다.
- **도움말**: 도움말이 필요하면 '?'키를 누른다.

맛보기 예제

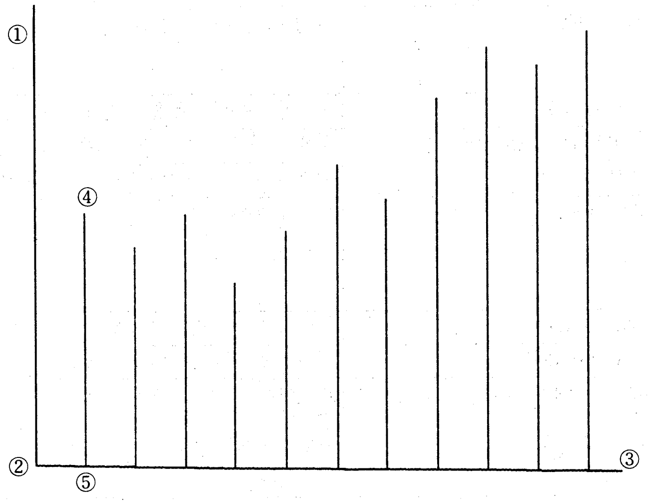
다음은 프로디자인을 이용하여 간단한 시트작성의 예를 그림과 함께 설명하겠다. 백문이 불여일견이라고 했듯이 한번 사용해보면 쉽게 이해를 할 수 있을 것이다.

먼저 라인기능에 대해서 알아 보기 위해 **그림 1**을 보기 바란다. ①, ②, ③지점에 포인트를 찍고 'V'키를 누른다. 참고로 'V'는 벡터(Vector)의 의미를 가지며 지정된 포인트간에 직선을 그려준다. (그려지는 순서는 지정된 포인트 순서에 따름) 그리고 포인트를 지정한다는 말은 앞서 설명했듯이 숫자 키패드의 '0'키를 누르면 되고 마우스를 이용하는 경우는 마우스의 왼쪽 버튼을 클릭시켜 주면 된다. 계속해서 ④번과 ⑤번, 그리고 ⑥번과 ⑦번을 같은 방법으로 포인트를 지정한 후 'V'키로 그려가면 그림 1과 같은 차트가 완성된다.

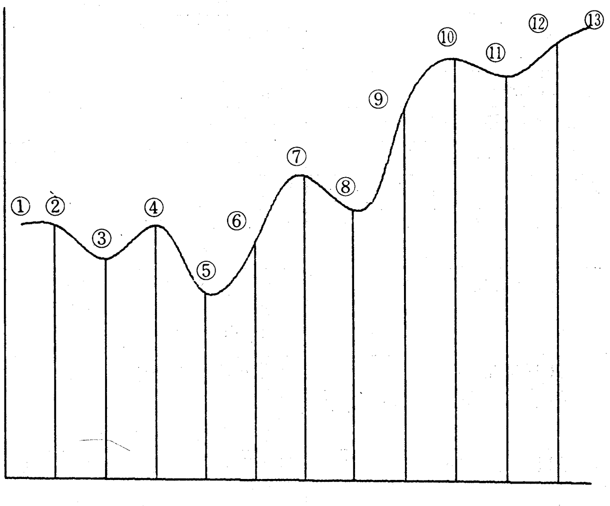
다음은 **그림 2**에서 지정된 포인트 ①~⑬을지정하고 'C'키를 눌러 보자. 그림 2에서 보는 바와 같이 각 꼭지점을 이어주는 곡선이 그려진다. 이 'C'는 커브(Curve)의 약자로 스프라인 곡선이라고 하는 곡선을 만들어 준다.

다음은 **그림 3**과 같이 원을 이용한 차트를 그려 보자. 그림 3에서와 같이 포인트 ①, ②를 지정하고 'O'키를 누른다. 그러면 화면에 포인트 ①을 중심으로 하고 포인트 ②를 반지름으로 하는 원이 그려질 것이다. 원이 그려지면 그림 1에서와 같은 방법으로 라인을 그린다.

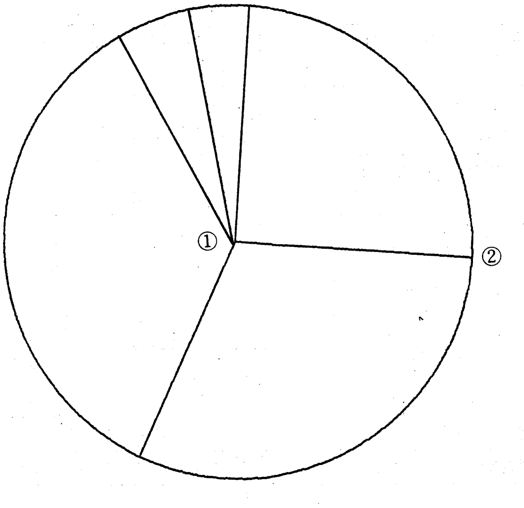
〔그림 1〕 직선 그리기



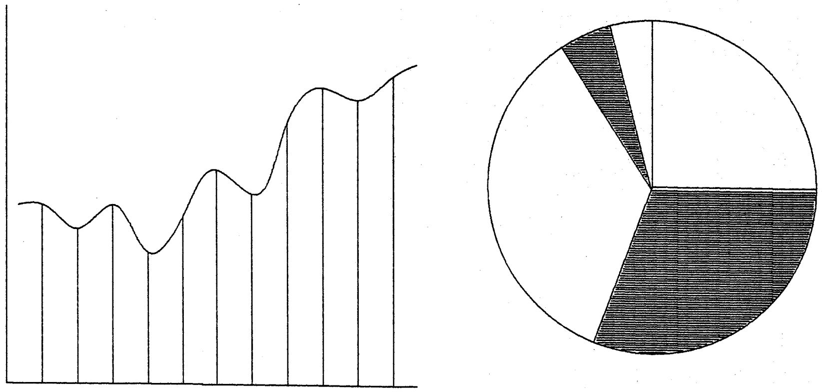
〔그림 2〕 곡선그리기



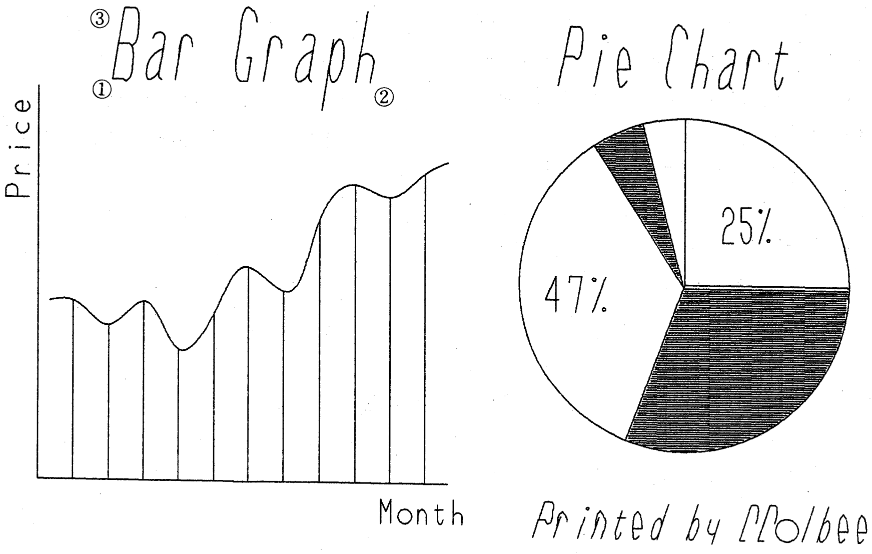
〔그림 3〕 원그리기



〔그림 4〕 색칠하기



〔그림 5〕 문자 입력



〔화면 2〕 프로디자인으로 설계한 키트 자동차

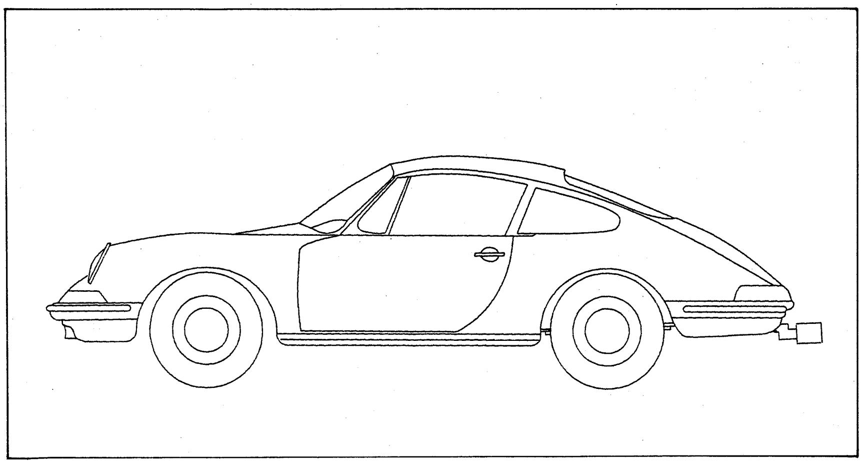


그림 4는 색칠을 하는 예를 보인 것으로 색칠을 할 부채꼴 모양 안에 아무 곳에서나 포인트를 선택하여 'P'키를 누른다. 'P'는 Paint의 약자이다.

마지막으로 **그림 5**는 텍스트를 작성하는 과정인데 포인트 ①, ②, ③을 차례로 지정하고 'T'키를 누른다. 그러면 화면 상단에 'Text:'라는 메시지가 나오는데 이곳에 원하는 텍스트를 입력하면 된다. 이 예에서는 'Bar Graph'라고 입력해 보자. 여기서 포인트 ①은 시작점을 말하며 포인트 ②는 끝점에 해당한다. 포인트 ③은 각도를 나타내는데 처음과 끝의 높이를 달리 하려면 ④번째 포인트를 끝점의 위치에서 원하는 곳에 지정한 후 문자를 입력하면 된다. 나머지 문자들도 마찬가지 방법으로 입력한다. 별다르게 어려운 것은 없으므로 몇분만 사용해 보면 금방 익힐 수 있을 것이다.

맺음말

프로디자인을 다른 관점에서 본다면 여기서 작성한 데이터를 다른 프로그램에서 응용하거나 심지어 한글의 벡터 폰트를 개발하는데도 유용하게 사용할 수 있을 것으로 본다. 또 한가지 짚고 넘어갈 것은 프로디자인에 약간의 버그가 숨어 있다는 말을 하고 싶다. 텍스트로 작성된 시트를 프린트 할 경우 글씨체가 원하는 대로 나오지 않는 경우가 있는데 이러한 경우에는 다른 폰트를 선택하여 출력해 보면 될 것이다. 매뉴얼을 보면 폰트가 1에서 9개까지의 화일이 있다고 했는데 필자가 가지고 있는 프로그램에는 2개뿐이다. 하여튼 이 버그만 제외한다면 프로디자인은 분명 독자들의 사랑을 받을 만한 좋은 캐드프로그램이 될 것이다. 끝으로 독자들이 직접 **화면 2**에 있는 멋있는 키트 자동차를 설계해 보기 바란다.

터보 C로 작성한 계산기 프로그램 PANTA

이형희/단국대 소프트웨어 연구회

여기에 소개하는 프로그램은 터보 C로 작성한 계산기 프로그램으로 제곱, 팩토리얼 계산 등의 다양한 계산식과 이러한 계산식의 결과를 화면에 그래프로 그려준다. 또한 공학도에게 꼭 필요한 적분이나 통계 등의 처리까지도 가능하며 무엇보다도 사용하기에 편리하다는 점이 커다란 장점이다.

프로그래밍 작업을 하거나 그외 다른일로 컴퓨터 앞에 앉아 있을 때 종종 계산기가 필요할 때가 있다. 보통은 베이직 인터프리터를 로드시킨 후 계산 결과를 얻거나 혹은 계산기(Calculator) 를 가지고 그때마다 문제를 해결했을 것이다.

전자의 경우는 제곱이나 팩토리얼계산 및 Deg상태에서의 삼각함수 계산이 비효율적이라는 문제가 있고 후자의 경우는 바로 앞에 ENIAC보다도 기능이 더 뛰어난 PC를 대하면서 또 계산기를 찾는 아이러니컬한 우를 범하는 꼴이 되어 버린다.

여기에 소개하는 PANTA.EXE는 이런 불편을 쉽게 해결해 줄 수 있을 뿐만 아니라 제곱, 팩토리얼계산 등 다양한 계산식을 해석하여 그 결과를 보여주는 프로그램이다. 더구나 형태로는 수학적 표현에 접근시켰으며 수식만 입력하면 그래프까지 그려주는 기능과 적분에서 통계까지 맡아 처리해주는 계산기 프로그램이다.

필자가 처음으로 계산기 프로그램을 생각했을 때 한가지 어려웠던 점은 입력되는 수식을 어떻게 해석하느냐 하는 문제였다. 우리가 흔히 계산수식을 기술할 때 '2*3-45'와 같이 항상 연산자 우선순위로 적는다는 것은 기대하기 어려운 일이고 실제로는 수식표현에 있어서 괄호를 많이 사용하며 괄호는 또 '(2*3*(-1)-3)'처럼 아무때나 나올 수 있는 성질의 것이어서 이러한 수식을 해석하는 일이 제일 먼저 선행되어야 했다.

이 문제는 연산자 우선순위법에서 발전한 L-R 주사법 혹은 L-R 파싱에 의해서 해결할 수 있었다. L-R문법은 Knuth에 의해 처음으로 고안된 알고리즘이나 최초의 것은 실용화될 정도로 효율이 좋은 것은 아니었다. 그후 SLR, LALR 등이 생겨 효율이 좋은 구문분석 생성이 가능하게 되었다.

L-R법의 동작은 다음과 같이 표현될 수 있다.

$$(S_0X_1S_1X_2S_2\cdots X_mS_m,\ a_i a_{i+1}\cdots a_n\ -1)$$

여기에서 S_i는 상태 $X_i \in V$를 나타내고 a는 입력기호를, $S_0X_1S_1X_2S_2\cdots X_mS_m$은 스택에 쌓여진 순서를(그림 1 참조), $a_ia_{i+1}\cdots a_n-1$은 아직 읽혀지지 않고 남아있는 입력기호를, 그리고 '−1'은 NIL을 나타낸다. 이에 대한 동작은 판독(read 혹은 shift)과 환원(reduce)으로 이루어지며 각각 다음과 같이 표현된다.

① 읽고나서 S로 천이한다. 이때의 스택 상태는 그림 1과 같다.

$$(S_0X_1S_1X_2S_2\cdots X_mS_m a_i S,\ a_{i+1}\cdots a_n\ -1)$$

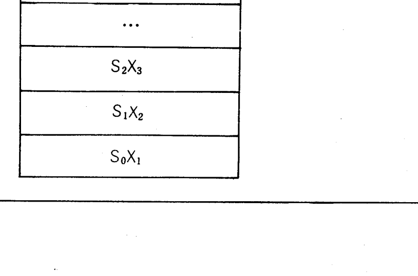
② A→α로 환원된다.

①단계에서 스택에 가장 최근에 들어온 것과 그다음 읽혀질 요소의 조합으로 결과를 구할 것인지, 아니면 스택에 그대로 집어 넣을 것인지를 판단한다. ②단계에서는 AS의 조합으로 처리결과를 구한 경우이다.

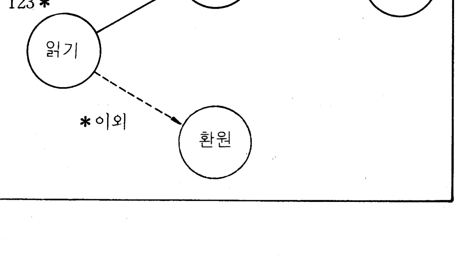
여기서 S_mA 는 123*과 같은 수식으로 볼 수 있다.

그림 2는 이와같은 관계를 알기쉽게 그림으로 나타낸 것인데 여기서 판독부

〔그림 1〕 L-R법에 이용되는 스택의 상태



〔그림 2〕 판독과 환원



분은 수치 다음에 연산자가 오는지의 여부를 판단하는 것이다. (틀리면 에러를 발생한다) 환원부분은 값이 한자리의 수치뿐만 아니라 123, 12, 57, 9999 등 여러자리를 가질 수 있음에 주목하기 바란다. 즉 연산자가 나올 때까지 반복하다가 한 단위의 수와 연산기호를 확인하는 과정을 나타내고 있는 것이다.

L-R주사법을 한마디로 표현한다면 식을 좌에서 우로 주사하면서 먼저 실행해서는 안되는 것은 스택에 쌓아놓고 다음에 읽어들인 연산자와 스택의 최상부에 있는 연산자와의 조합에 의해 다음 동작(스택에 쌓아 놓을까 혹은 꺼내서 처리할까)을 결정하는 방법이다.

다음에는 PDL법에 의한 알고리즘을 소개하겠다. 이번에는 실제 예제와 함께 리스트 1의 PANTA.C를 참조하기 바란다. PDL법의 알고리즘은 다음과 같이 표현될 수 있다.

입력 : $a_1a_2\cdots a_n$NIL→PANTA.C의 Keyin()부분

초기설정 : 스택에 NIL을 놓는다.

J : =0;

입력기호 a_j를 읽는다.→interpreter()부분

while 스택의 top에 있는 연산자(S_i)와 → stackcal()부분

현재의 입력기호 a에 대하여

a_i)a_j가 성립한다.

d_o 스택에서 꺼낸다.

if a_i=NIL then 끝

else a_j를 스택에 넣는다.

반복부분 끝

(α)

여기서 @ 부분에 해당하는 루틴을 PANTA.C에서 참조하면 다음과 같다.

```
if (_tell(jj) >= _tell(s2) &&
        (s3 == kk)) {
    t=_s_calcul(s1,s2,hh);
    push(t,jj,s3);
    if (stack == 1) {
      if (_gl_err != 0) return -1;
      else return 0;
      }
    }
    else
    {
      push(s1,s2,s3);
      push(hh,jj,kk);
      return 0;
    }
  }
  return -1;
}
```

〔표 1〕 a*b+c의 처리과정

수식 : a*b+c

단계	처 리	스택의 내용
1	a, *를 스택에 집어 넣는다.	스택의 내용은 a, *이다.
2	b, +를 만난다. a, *를 꺼낸다	없음
3	*가 +보다 연산우위이므로 a*b→T, +	없음
4	T, +를 스택에 집어 넣는다.	스택의 top은 T, +
5	C와 NIL을 만난다. T, +를 스택에서 꺼낸다.	없음
6	결과는 T이다.	

〔표 2〕 a+b*c의 처리과정

수식 : a+b*c

단계	처 리	스택의 내용
1	a를 만난다	없음
2	+를 만나 a, +를 스택에 집어 넣는다.	a, +
3	b, *를 만난다. a, +를 스택에서 꺼내어 체크한다.	없음
4	*가 +보다 연산우위이므로 a, +와 b, *순으로 스택에 넣는다.	스택의 top b top+1 * top+2 a top+3 +
5	C와 NIL을 만난다.	스택의 top b top+1 * top+2 a top+3 +
6	스택에 넣은 역순으로 꺼낸다. b*c→T, NIL	스택의 top a top+1 +
7	T+a→T, NIL	없음
8	결과는 T이다.	

표 1, 2는 이 알고리즘에 의해 2개의 수식 'a*b+c'와 'a+b*c'을 예로 처리과정을 보인 것이다.

그러나 이러한 연산자 우선순위법은 +, −, *, / 의 사칙연산에만 적용되며 모든 수식에 적용시킬 수 없으므로 여기서는 이를 기본으로 알고리즘을 확장시켰다. (여기서 모든 수식이라고 한 것은 sin(, cost, ln(의 함수 형태가 포함된 수식을 말하는 것이다.) 더우기 이 프로그램에서는 제곱(n^2), 팩토리얼(!), 제곱근($\sqrt{\ }$),n제곱근($\sqrt[n]{\ }$), 등이 들어가므로 앞에서 언급한 내용과 더불어 식의 해석이 더욱 힘들게 된다. 여러 궁리 끝에 필자의 경우는 모든 연산자를 3종류로 분류하게 되었다(그림 3 참조) 이러한 기본적인 분류에 의해서 수식의 해석이 가능하게 되었고

수학적인 표현에 어느 정도 근사한 계산기를 만들 수 있었다.

프로시듀어 설명

getkey( )

터보 C의 함수 bioskey에 의해 Caps-key 체크와 특수 확장키의 값을 265 이상의 값으로 하여 되돌려준다. #define 문에서 NO_BEEP가 있는 라인을 삭제해 버리면 키를 누를 때마다 삑삑하는 음을 들 수 있다. 그리고 프로그램의 선언부에서 #define NO_-BEEP 를 일부러 집어넣은 이유는 이 문을 안썼을 때 일부 4.77MHz XT에서 'system halt'에러가 발생하기 때문이다.

• set_hur( )

6845 레지스터를 조작하여 640✱400의 해상도를 갖는 화면으로 만든다.

• set_cga( )

CGA는 IBM-PC의 표준이므로 롬바이오스(ROM-Bios)를 이용한다. 640✱200 의 해상도로 세트할 수 있으며 팔레트(Pallette)는 3으로 만든다. 즉 이 프로그램은 CGA나 허클리스 보드에서 사용할 수 있다.

• set_text( )

그래픽모드에서 텍스트모드로 전환한다.

• range( )

그래픽모드에서 한눈금(격자)의 길이를 지정한다.

• main( )

레지스터 AH에 15를 대입한 후 Bios 10H를 실행시킨다. 그래서 변화되는 AH값을 가지고 그래픽 보드의 종류를 판단하고 변수의 초기화와 대략적인 화면의 구성을 한 후 제어를 프로시듀어 keyin( )으로 넘긴다.

• control_cha( )

모든 확장키와 콘트롤 키를 처리한다.

• keyin( )

모든 키의 처리를 담당한다. 참고로, 변수 mag는 현재의 커서 위치에서 수식의 끝까지의 길이를 가지며 _ma_sto는 수식의 끝을 나타내는 값이고 jump_condition은 워크시트에서 스크롤(한 행 위로)의 조건을 판단한다.

• interpreter( )

모든 수식의 해석에 핵심적인 부분으로 변수 minus_fl은 함수의 '+', '−'의 여부를 판단하고, stack은 현재 스택에 몇개의 수식이 쌓여있는지를, dep는 괄호의 갯수를 기억하고 set는 숫자의 경우는 '0'의 값을, 연산자의 경우는 '1'의 값을 기억하게 된다. 그외 t와 _glo_ans는 계산결과를 기억하고 jum는 계산결과를 표시할지의 여부를 판단하며 _buffering[390]은 계산할 수식을 기억하고 있다.

• tell( )

연산처리의 우선순위를 판단하여 알려준다.

• searfun( )

함수를 찾는다.

• stackcal( )

스택에서 연산자 우선순위대로 계산을 한다.

• state_list( )

현재의 계산기 상태(Deg, Rad, Caps, Insert, Over)를 나타낸다.

• sigma( )

통계부분에 쓰인다.

• integral( )

적분부분에 사용되며 Trepzodial rule을 적용시킨다.

• graph( )

그래프를 그려주는 부분이다. BOAD의 값이 B800이면 Cga, 아니면 Hur이다. _x_integer는 x눈금의 간격을, _y_integer는 y눈금의 간격을 나타낸다.

• _o_s_shell( )

Dos Shell부분으로 'exit'로 복귀한다.

• run( )

_area_buff에 있는 내용을 실행하고 결과를 얻는다.

• area( )

_area_buff로 수식을 입력하는 부분이다.

한편 이 프로그램에서 쓰인 광역변수(Global Variable)는 다음과 같다.

• _gl_err

에러가 발생했을 때 어떤 종류의 에러인가를 판단한다. 에러가 없을 때는 '0'값을 지니게 된다.

• mode

현재 화면의 모드(Mode)상태를 기억하고 있다.

• convert_to_deg

라디안(Radian)값을 디그리(Degree) 값으로 변환할 때 이용된다.

• _in

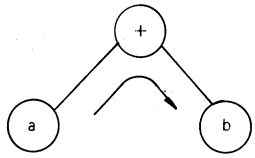
삽입모드(Insert) 상태인지 아닌지(Overwrite)를 나타낸다.

• _deg

디그리값인지 라디안값인지를 판단한다.

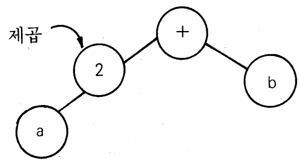
• glo_i

〔그림 3〕 3종류로 분류된 연산자



예) a+b
즉 연산자 양쪽에 변수가 오는 연산자
(+, −, ✱, /, $\sqrt[n]{\ }$( ))

(a) 일반적인 연산자



예) a^2+b
1✱−1
(², !, −)
✱여기서 '−'는 ⓐ의 연산자 다음에 오는 '−'이다.

(b) 제곱과 같이 한번의 과정을 거쳐야 (a)의 형태와 같아지는 연산자

제곱 ($\sqrt{\ }$(, sin(, log()

(c) 괄호로 시작해서 괄호로 끝나는 연산자

_buffering의 데이터를 스캔할 때 변하는 수이다.

• listfun

수식에 쓰여진 함수이름을 지니고 있다.

프로그램 실행환경 및 이용법

리스트 1의 소스리스트를 가지고 512KB 이상의 메모리에서 터보C(버전 1.0이상)로 컴파일 한다. 실행화일의 길이는 약 47KB가 되며, CGA와 허큘리스 보드 모두를 지원할 수 있다.

사용변수는 a~z까지 26개가 있고 여기서 F1키는 변수의 값을 변화시키고, F2키는 그 내용을 볼 수 있게 해준다. 상수는 디폴트로 π가 마련되어 있고, 사용 연산자 및 함수는 표 3에 보인 바와 같다. 그리고 여기서는 연산자 우선 순위는 다음과 같은 순서로 정했다.

① ^{2}, !
② 함수
③ Y^n, $\sqrt{\ }$, $\sqrt[n]{\ }$
④ ∗, /, +, −, %, −pol

DOS 프롬프트 상태에서 직접 'PANTA' 를 입력시켜 실행시키면 화면 1과 같이 워크시트(WorkSheet), 에러메시지(Error message), 메뉴(Menu), 상태표시(State)의 윈도우가 나타나고 좌측하단에는 기본적인 키 조작법이 나타난다. 이들의 키기능을 잠시 살펴보면 다음과 같다.

●^F : 〈CTRL〉-F

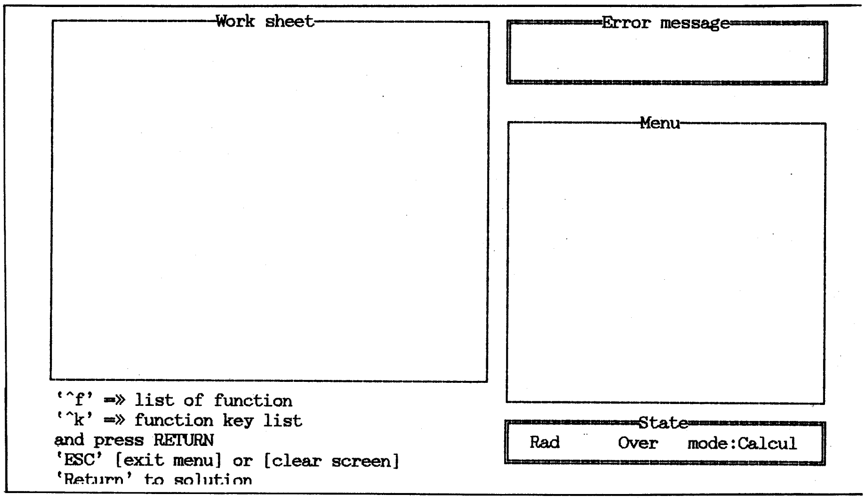
이 프로그램에서 사용할 수 있는 사용함수를 보여준다. 일단 메뉴가 나오면(화면 2 참조), ←, →, ↑, ↓키를 이용하여 원하는 항목을 선택하고 리턴키를 치면 선택항목이 실행된다.

① Drg: 삼각함수의 계산값을 디그리나 라디안값으로 변환한다.
② CLEAR: a~z까지의 모든 변수를 '0'으로 만든다.
③ Σ: 기초통계에 이용된다.
④ integ: 적분에 이용된다.
⑤ Bin: 입력한 수를 2진수로 바꾼다.
⑥ Hex: 입력한 수를 16진수로 바꾼다.
⑦ Oct: 입력한 수를 8진수로 바꾼다.
⑧ 기타 : 익히 알려진 함수들

●^K : 〈CTRL〉-K

이번엔 함수키들의 리스트가 메뉴

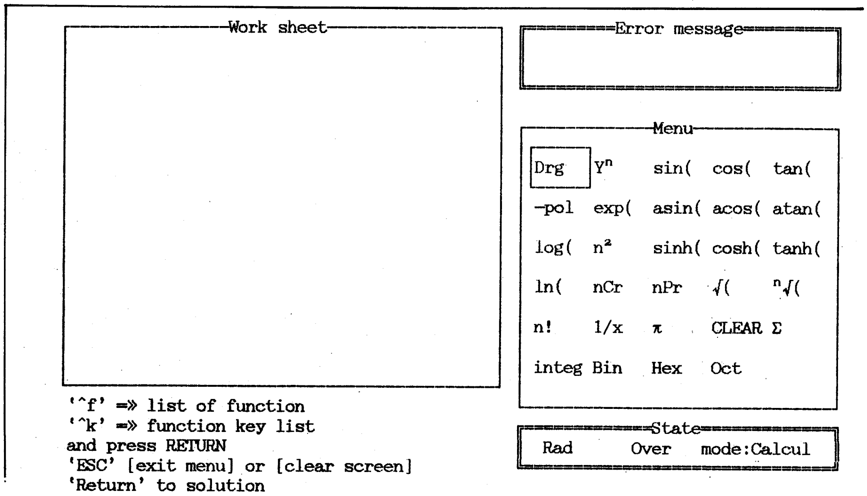
〔화면 1〕 초기실행화면



〔표 3〕 사용가능한 연산자 및 함수

연 산 자	함 수
² : 제곱	sin(
Yⁿ : 거듭제곱	cos(
−Pol : Polynomial근	tan(
% : 나머지	asin(
P : Permutation	acos(
C : Combination	aton(
! : Factorial	sinh(
1/ : 역수	cosh(
√‾ : 제곱근	tanh
ⁿ√‾ : n제곱근	exp(
	log(
	ln(
	√‾(
	ⁿ√‾(

〔화면 2〕 콘트롤-F의 화면



윈도우에 나타난다(화면 3 참조).

① FUNC: 사용가능한 함수를 보여준다.
② GRAPH: 그래픽으로 함수를 그려 준다.
③ Shell: DOS SHELL기능을 수행한다.
④ Area: area모드로 계산 수식을 입력한다.
⑤ Store(F1): a~z까지의 변수에 수를 기억시킨다.
⑥ Rec(F2): a~z까지의 변수에 기억시킨 변수를 보여준다.
⑦ Run(F3): 프로그램 모드에서 입력한 수식의 값을 묻고 계산하여 준다.
⑧ Range(F4): 그래픽 부분에서의 Range 이용
⑨ Quit(F5): 프로그램 종료
⑩ Dir(F6): 디렉토리를 보여준다.
⑪ Load(F7): 계산할 수식을 로드한다.
⑫ Save(F8): 수식을 화일로 기억시킴.

표 4는 예제로서 일반적인 수식들을 입력하는 방법과 그 결과들이 어떻게 나왔는지를 보인 것이다. 참고로 통계 수식을 처리할 때 입력되는 데이터가 끝나면 ESC키를 누르는 것과 최대 40개까지의 함수 사용이 가능하고 대수식의 기억용량이 390 스텝까지 가능하다는 것을 알아두기 바란다.

화면 4는 Graph(^G)를 선택해서 원하는 함수의 그래프를 보이고 있다. 메뉴에서 Graph를 선택하면 워크시트에 다음과 같은 메시지가 나온다.

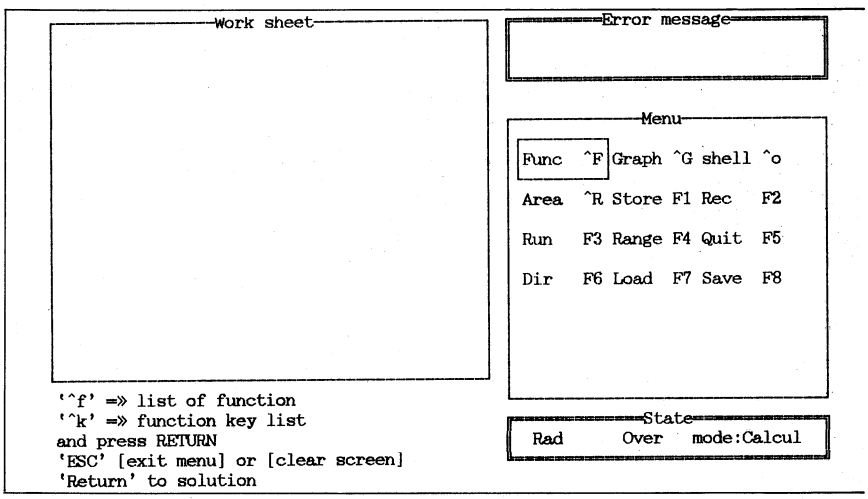
Formula f(a)=

여기에 'a²+a+a1', 'a*sin(a)' 등 원하는 수식을 입력하면 해당하는 그래프 출력을 얻을 수 있다.

이번 프로그램을 작성하면서 가장 절실히 느낀 점은 전산학의 비전공인으로서 주로 컴파일러 이론인 전산학 이론을 익히는데 있었다.

그러므로 전산화의 비전공인들은 먼저 기본적인 전산학이론을 학습하고 나서 프로그램을 작성하라고 권하고 싶은게 필자의 반응이다. PC

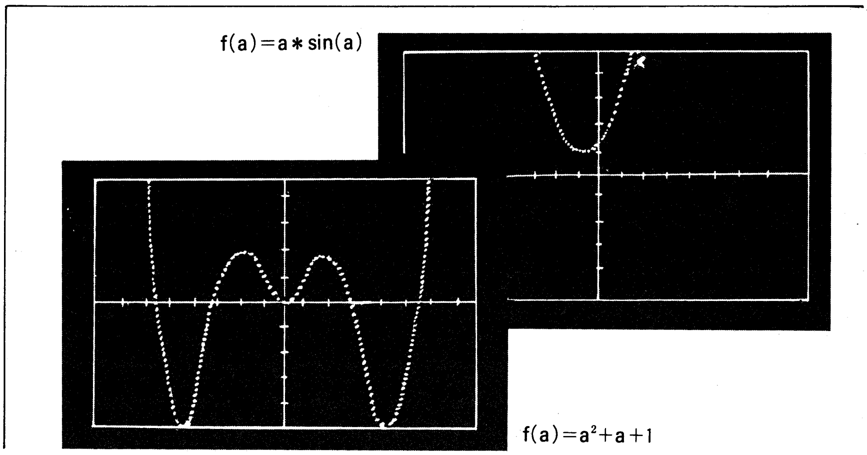
〔화면 3〕



〔표 4〕 일반적인 수식의 표현과 결과

수식	입력	결과
$\sqrt{2^2+4^2}$	2-pol4	4.47213595499957872
2^2	2²	4
3.14	3.14159265358979	3.14159265358979001
1.2^3 (누승)	1.2Y^n3	1.72799999999999998
$\sqrt{3}$	√(3)	1.73205080756887742
$\sqrt[3]{1.2}$	3^n√(1.2)	1.06265856918261115
90!	90!	1.48571596448176069e+138
a=1	1 Sto a	1

〔화면 4〕 그래프 출력 예



〔리스트 1〕 PANTA.C 리스트

1

```c
#include <stdio.h>
#include <dir.h>
#include <dos.h>
#include <math.h>
#include <process.h>
#include <stdlib.h>

#define _ESC     27
#define RET      13
#define Cont_g    7
#define UP      328
#define DOWN    336
#define RIGHT   333
#define LEFT    331
#define Cont_f    6
#define Cont_k   11
#define Cont_o   15
#define Cont_r   18
#define _f1     315
#define _f2     316
#define _f3     317
#define _f4     318
#define _f5     319
#define _f6     320
#define _f7     321
#define _f8     322
#define _del    339
#define _bs       8
#define _ins    338
#define SQRT     -5
#define SQR      -3
#define POL     -60
#define NSQ      -4
#define NO_BEEP   ← 이 부분을 제거하면 키를 누를 때마다 CLICK 음을 들을 수 있음
int _glo;      /*integer variavle define*/
int glo_i,j,k,_ma_x,_ma_y,_ma_sto,_m_x,_m_y;
int listfun[40];
int _mem_sto;
int mag;
int mode;
int cps;
int _gl_err;
char _file[15];
double convert_to_deg;
double glo_double;

struct ss {
        double li;
        char ch;
        int fn;
```

2

```c
        };

char _str1[30][6]={"Drg  ","Y    ","sin( ","cos( ","tan( ",
                   " pol ","exp( ","asin(","acos(","atan(",
                   "log( ","n    ","sinh(","cosh(","tanh(",
                   "ln(  ","nCr  ","nPr  "," (   ","  (  ",
                   "n!   ","1/x  ","     ","CLEAR","     ",
                   "integ","Bin  ","Hex  ","Oct  ","     "};

char _str2[12][ 9]={"Func  ^F","Graph ^G","shell ^o",
                    "Area  ^R","Store F1","Rec   F2",
                    "Run   F3","Range F4","Quit  F5",
                    "Dir   F6","Load  F7","Save  F8"};

char *ch_state[] = {"Rad","Deg",
                    "Caps",
                    "Insert","Over",
                    "Calcul","Graph","Program"};

struct ss stackli[255];
int kk,s3,c3,fi,j,k,stack,yo,dep;

char MUL='*',ADD='+',DIV='/',YYY='Y';

char gch,jc,s2,c2,jj;
int stack,_deg=0,_in=1,_st=0,caps=1;
unsigned int BORD;

char _buffering[390],_area_buff[390];

double _x_inter = 1.;
double _y_inter = 1.;
double load[26];
double _glo_ans,t;
double c1,s1,hh;

void _box(int,int,int,int,char print_string[],int,int),
     _cls(int,int,int,int);
void _scroll_up();
void _item_pro(int),_w_s_pr();
void _error_print(int);
void pop(),push(double,char,int);
void _dir();
void next(int),_o_s_shell();
void _graph(),_back();
void putcc(int,int,int),integral();
void _bin(double),_hex(double),_oct(double);
void _dec(double,int),video_string(int ,int ,char STRING_DATA[]);
void _run(),load_vi();
void area(),save_vi();
void cls_work_sheet(),_state_list();
```

3

```c
double _atof();
double _s_calcul(double ,char ,double);
double _squre(double),_n_pol(double,double);
double _n_root(double,double),_n_squre(double,double);
double _n_p_fac(double,double),_mod(double,double);
double _n_c_fac(double,double),_n_fac(double);
double _cal_fun(int,double),_tri_cal(int,int,double);
double four_operation(double ,char __oper ,double );

int _tell(int),_sigma();
int keyin(int),_scanf2(int , int , char *);
int _scanfd(int , int , double *);
int _interpreter(int),chrcmp(char f_string[],char s_string[]);
int _control_cha(int , unsigned int);
int _stackcal(),_leng(double);

gotoxy(x,y)
int x,y;
{
   union REGS r;
   r.h.ah = 2;
   r.h.dl = x;
   r.h.dh = y;
   r.h.bh = 0;
   int86(0x10,&r,&r);
}

getkey()   ← 확장키는 256이상으로 값을 돌려주고 표준키는 그 고유값을 돌려준다.
{
   unsigned int sound,i;
   #ifdef NO_BEEP
   #else
      union {
            long divs;
            unsigned char c[2];
         } con;
   #endif
   unsigned char g;

   sound=bioskey(0);
   #ifdef NO_BEEP
   #else
    con.divs=1193280/sound;
    outport(67,182);
    outport(66,con.c[0]);
    outport(66,con.c[1]);
    g=inportb(97);
    outport(97,g | 3);
    for (i=0 ; i <= 1000; ++i);
    outport(97,g);
   #endif
```

```
   _state_list(2);
   if ((sound & 0x00ff) == 0) {
      sound=sound >> 8;
      sound += 256;
      return sound;
   }
   return sound & 0x00ff;
}

void _set_hur()◀── 허큘리스의 640×400
{
   int  _set_ta[12]= {0x31,0x28,0x29,
                      0x08,0x68,0x02,
                      0x64,0x65,0x02,
                      0x03,0x02,0x01};
   int i;
   outport(0x03bf,1);
   outport(0x03b8,2);
   for (i = 0; i <= 11; i++) {
      outport(0x03b4,i);
      outport(0x03b5,_set_ta[i]);
   }
   outport(0x3b8,10);
}

void _set_cga()◀── CGA의 640×200을 지원
{
   union REGS r;
   r.h.al=6;
   r.h.ah=0;
   int86(0x10,&r,&r);
   r.h.bh=1;
   r.h.bl=3;
   r.h.ah=11;
   int86(0x10,&r,&r);
}

void _set_text()◀── 텍스트모드로 전환
{
   union REGS r;
   r.h.al=3;
   r.h.ah=0;
   int86(0x10,&r,&r);
}

void plot(x,y) ◀── 화면에 점을 찍는다.
int x,y;
{
   unsigned off_set;
   char bitd,bitn;
   off_set=0x2000*(y%4)+80*(int)(y/4)+(int)(x/8);
```

4

```
   bitn=1 << (7-x % 8);
   bitd = peekb(BORD,off_set);
   bitd = bitd | bitn;
   pokeb(BORD,off_set,bitd);
}

void clear_graphic() ◀── 그래픽 화면을 지운다.
{
   unsigned int i;
   for (i=0; i <= 0x7fff; ++i) pokeb(BORD,i,0);
}

void _range()──┤ 그래프의 (x, y) 좌표에
               │ 대응하는 수치를 변경할 때
{
   char _str[20],ch;
   int _mem_ma;
   _mem_ma=_ma_y;
   _ma_y=7;
   _cls(48,7,78,19);
   video_string(52, 8,"X Range:");
   video_string(52,10,"Scale  [old]:");
   _ma_y= 9; _dec(_x_inter,67);
   video_string(52,11,"Scale  [new]:");
   video_string(52,13,"Y range:");
   video_string(52,14,"Scale  [old]:");
   video_string(52,15,"Scale  [new]:");
   _ma_y=13; _dec(_y_inter,67);
   if (_scanfd(65,11,&_x_inter) < 0) {
      _ma_y=10;
      _dec(_x_inter,68);
   }
   if (_scanfd(65,15,&_y_inter) < 0) {
      _ma_y=14;
      _dec(_y_inter,68);
   }
   _ma_y=_mem_ma;
}

int _sel_term(_lx,_ly,_size,_term)─┤ 화면의 메뉴부분에서
int _lx,_ly,_size,_term;           │ 항목선택시 사용
{                                  │ (ESC)로 복귀한다.
   int i,j,k;
   int _cc;
   i=48; j=7; k=0;

repeat:
   _box(i,j,(i+_size+1),j+2,NULL,1,0);
   _cc=getkey();
   _era_box(i,j,(i+_size+1),j+2);
jum:
```

5

```
switch (_cc) {

    case _ESC    : _cls(49,7,78,17);
                   return(-1);

    case LEFT    : i -= (_size+1);
                   if (i < 48) {
                      if (j > 7) i=48+(_lx-1)*(_size+1);
                      else
                      {
                        i += (_size+1);
                        goto repeat;
                      }
                      k += (_lx-1);
                      _cc=UP;
                      goto jum;
                   }
                   if (k == 0) goto repeat;
                   --k;
                   break;
    case RIGHT   : i += (_size+1);
                   if (((_lx-1)*(_size+1)+48) < i) {
                      i = 48;
                      k -= (_lx-1);
                      _cc=DOWN;
                      goto jum;
                   }
                   ++k;
                   if (k == _term) {
                      --k;
                      goto repeat;
                   }
                   break;
    case UP      : j -= 2;
                   if (j < 7) {
                      j += 2;
                      goto repeat;
                   }
                   k -= _lx;
                   break;
    case DOWN    : k += _lx;
                   if ((j+1) > ((_ly-1)*2+7)) {
                      k -= _lx;
                      goto repeat;
                   }
                   if (k > _term) {
                      k -= _lx;
                      goto repeat;
                   }
                   j += 2;
                   break;
```

6

/PANTA/

7

```
      case RET     : return(k);
   }
   goto repeat;
}
void video_string(x,y,string_data)── 화면에 문자열을 인쇄
int x,y;
char string_data[];
{
   int i;
   for (i=0 ; (gch=string_data[i]) != '\0';++i)
       pokeb(BORD,(x+i)*2+y*160,gch);
}

void cls_work_sheet()── 워크시트 부분을 지운다.
{
   _cls(2,2,44,18);
}

main ()
{
   int i;
   union REGS r;
   char GAL=']';
   r.h.ah=15;                               ┐
   int86(0x10,&r,&r);                       │
   if (r.h.al == 7) BORD=0xb000;            ├ 그래픽보드의 종류를
   else BORD=0xb800;                        │ 결정하고 화면을 지운다.
   _set_text();                             │
   _gl_err=0;                               ┘
   _cls(1,1,25,80);

   video_string(2,20,"'^f'"); putcc(7,20,205); putcc(8,20,175);
   video_string(10,20,"list of function");
   video_string(2,21,"'^k'"); putcc(7,21,205); putcc(8,21,175);
   video_string(10,21,"function key list");
   video_string(2,22,"'f5' quit");                    HELP
   video_string(12,22,"select[");                     부분을
   putcc(19,22,24); putcc(20,22,25);                  그린다.
   putcc(21,22,26); putcc(23,22,27);
   putcc(24,22,GAL);
   video_string(26,22,"and press RETURN");
   video_string(2,23,"'ESC' [exit menu] or [clear screen]");
   video_string(2,24,"'Return' to solution");

   _box( 1, 1,45,19,"Work sheet"   ,10, 0);  ┐
   _box(47, 1,79, 4,"Error message",13, 1);  │ 화면의 박스
   _box(47, 6,79,20,"Menu"         , 4, 0);  │ 부분을 지운다.
   _box(47,21,79,23,"State2"       , 5, 1);  ┘

   _str1[ 1][1]=252; _str1[11][1]=253; _str1[22][0]=227;
```

8

```
   _str1[18][0]=251; _str1[19][0]=252; _str1[24][0]=228;
   _str1[ 5][0]=196; _str1[19][1]=251;

   for (i=1; i <= 4; ++i)── 계산기의 상태표시
      _state_list(i);
   for (i=0; i <= 26;++i)  load[i]=0.0;
   convert_to_deg=90./asin(1); ←── rad값을 deg로
   _area_buff[0]='\0';              변환할 때의 수치계산
jum:
   if (keyin(0) < 0) { ←── 제어를 keyin으로 넘긴다.
      _cls(1,1,80,25);
      gotoxy(1,1);
      _set_text();
      exit(0);
   }
   goto jum;
   }

   void next(plus_condi)── 워크시트에서 커서를 우로 옮긴다.
   int plus_condi;
   {
      ++_ma_x;
      if (_ma_x > (_m_x+42) && plus_condi == 0) {
         ++_ma_y;
         _ma_x=2;
      }
      if (_ma_y > 18) {
         _scroll_up();
         _ma_y=18;
      }
      gotoxy(_ma_x,_ma_y);
   }

   void _next(plus_condi)
   int plus_condi;
   {
      ++_ma_x;
      if (_ma_x > (42+_m_x) && plus_condi == 0) {
         ++_ma_y;
         _ma_x=2;
      }
      if (_ma_y > 18) {
         _scroll_up();
         _ma_y=18;
      }
   }

   int _control_cha(soon,ch)── 모든 확장키와 콘트롤키를 담당
   int soon;
   unsigned int ch;
```

9

```
{
   int _fun,set=0,i,j;
   int _mem_x,_mem_y,_mem_j;
   char copy_buffering_data[390];
   char ab,dat[40]={"0000000000000000000000000000000000000000"};
   FILE *fp,*fopen();
rep:    /* repeat switch stat */
   switch (ch) {

        case LEFT   : --_ma_x;
                      if (_ma_x < _m_x) {
                         if ((_ma_y-_m_y) > 0) {
                            _ma_x=_m_x+42; --mag;
                            --_ma_y; --_ma_sto;
                            gotoxy(_ma_x,_ma_y);
                            return 0;
                         }
                         else
                         {
                            ++_ma_x;
                            return 0;
                         }
                      }
                      --mag;
                      --_ma_sto;
                      gotoxy(_ma_x,_ma_y);
                      return 0;
        case RIGHT  : ++_ma_x;
                      ++mag;
                      if (mag > 0) {
                         mag=0;
                         --_ma_x;
                         return 0;
                      }
                      if (_ma_x > 44) {
                         _ma_x=2;
                         ++_ma_y;
                      }
                      ++_ma_sto;
                      gotoxy(_ma_x,_ma_y);
                      return 0;
        case Cont_f : _item_pro(0);
                      _fun=_sel_term(5,6,5,31);
                      _cls(48, 7,78,18);
                      if (_fun < 0) return -1;
                      if (_fun == 0) {
                         _state_list(1);
                         return 0;
                      }
                      if (_fun <= 22) {
                         _w_s_pr(_fun,30);
```

```
           return 0;
        }
        ++_ma_y;
        _ma_x=2;_m_y= (++_ma_y);
        if (_ma_y > 18) {
           _scroll_up();
           _ma_y=18;
        }
        if (_fun > 22)  {
           if (t == 0.0 &&
               _fun != 24 && _fun != 25 && _fun != 23) {
              j=_interpreter(1);
              if (j < 0) {
                 _m_y = (--_ma_y);
                 return -1;
              }
           }
           switch (_fun) {
                  case 23 : for (i=0; i <= 26; ++i)
                                   load[i]=0.0;  break;
                  case 24 : _mem_x=_m_x; _mem_y=_m_y;
                            _sigma();
                            _m_x=_mem_x; _m_y=_mem_y;
                            cls_work_sheet(); return -3;
                  case 25 : integral();
                            _m_x=2;  _m_y=2;   return -2;
                  case 26 : _bin(t);           break;
                  case 27 : _hex(t);           break;
                  case 28 : _oct(t);           break;
           }
           ++_m_y;
           if (_m_y > 18) {
              _scroll_up();
              _m_y=18;
           }
           return -3;
        }
        return(0);
case _ESC   : cls_work_sheet();
        _cls(49,2,78,3);
        _in=1;
        mag=0;
        _ma_sto=-1;
        _m_x=2; _m_y=2;
        return(-2);
case Cont_k : _item_pro(1);
        _fun=_sel_term(3,4,8,12);
        _cls(48, 7,78,18);
        if (_fun < 0) return -1;
        switch(_fun) {
              case  0 : ch = Cont_f; goto rep;
```

10

```
                 case  1 : ch = Cont_g; goto rep;
                 case  2 : ch = Cont_o; goto rep;
                 case  3 : ch = Cont_r; goto rep;
                 case  4 : ch = _f1;    goto rep;
                 case  5 : ch = _f2;    goto rep;
                 case  6 : ch = _f3;    goto rep;
                 case  7 : ch = _f4;    goto rep;
                 case  8 : ch = _f5;    goto rep;
                 case  9 : ch = _f6;    goto rep;
                 case 10 : ch = _f7;    goto rep;
                 case 11 : ch = _f8;
              }
              goto rep;
case Cont_g : _st=1; _state_list(4);/ call graph  /
              _mem_x=_m_x; _mem_y=_m_y;
              _graph();
              _st=0; _state_list(4);
              _m_x=_mem_x; _m_y=_mem_y;
              cls_work_sheet();
              return -3;
case Cont_o : _o_s_shell();
              _m_x=2; _m_y=(++_ma_y);/ call dos-shell  /
              return -3;
case Cont_r : _st=2; _state_list(4);/  run area  /
              area();
              _st=0; _state_list(4);
              _m_x=2;
              return (-2);
case _f1    : if (soon != 0) break;
              ++_ma_sto;
              _buffering[++_ma_sto]=' ';
              next(0);
              putcc(_ma_x,_ma_y,32);
              for (i=0 ; i <= 2; ++i) {
                 next(0);
                 putcc(_ma_x,_ma_y,_str2[4][i]);
                 _buffering[++_ma_sto]=_str2[4][i];
              }
              next(0); putcc(_ma_x,_ma_y,32);
              ++_ma_sto;
     ban    :
              next(0);
              ab=getkey();
              if (ab == _ESC) {
                 _m_x=2;
                 _m_y=2;
                 return (-2);
              }
              if (ab < 'a' || ab > 'z') {
                 --_ma_x;
```

11

```
              goto ban;
           }
           putcc(_ma_x,_ma_y,ab);
           _buffering[++_ma_sto]=' ';
           next(0);
           putcc(_ma_x,_ma_y,32);
           _buffering[++_ma_sto]=ab;
           _buffering[++_ma_sto]=RET;
           if (_interpreter(0) < 0) {
              goto ban;
           }
           load[ab-'a']=t;
           _m_y = _ma_y;
           _m_x = 2;
           if (_ma_y > 18) _scroll_up();
           return -3;
case _f2    : ab = getkey();
           set=0;
           if (ab < 'a' || ab > 'z') return 0;
           gcvt(load[ab-'a'],40,dat);
           for (j=37; j >= 1; --j)
              if (dat[j] != '0') {
                 set = 1;
                 _mem_j= j;
                 break;
              }
           if (set != 1) break;
           if ((_ma_x+j) > 44) {
              ++_ma_y;
              _ma_x=(_ma_x+j)-43;
              if (_ma_y > 18) {
                 _scroll_up();
                 _ma_y=18;
              }
            }
            else _ma_x=_ma_x+j;
            for (i=_mem_j; i >= 0 ; --i) {
                if (_ma_x < 2) {
                   if (i == 0) goto jum3;
                   _ma_x=43;
                   --_ma_y;
                   if (_ma_y > 2) {
                      _ma_x=2;
                      _ma_y=2;
                   }
                }
                gotoxy(_ma_x,_ma_y);
                _buffering[_ma_sto+i]=dat[i];
                putchar(_buffering[_ma_sto+i]);
                --_ma_x;
           }
```

12

13

```
jum3        :
              _ma_x += _mem_j;
              next(0);
              _ma_sto=(_ma_sto+_mem_j);
              return 0;
case _f3    : _run(); _m_x=2;    return -2;
case _f4    : _mem_x=_m_x; _mem_y=_m_y;
              _range();
              _m_x=_mem_x; _m_y=_mem_y;
              _cls(49, 7,78,18); return -2;
case _f5    : return -1;
case _f6    : _mem_x=_ma_x; — dir 리스트
              _mem_y=_ma_y;
              _dir();
              _ma_x=_mem_x;
              _ma_y=_mem_y;
              _m_x=2;
              return -2;
case _f7    : _mem_x=_ma_x; _mem_y=_ma_y;
              if (_load() < 0) {
                 _m_x=2;  _m_y=2;
                 _cls(49,7,78,18);
                 return -2;
              }
              _cls(49,7,78,18);
              mag=0;
              return 0;
case _f8    : _cls(49,7,78,18);
              video_string(49,7,"Save filename:");
              for (i=0; i <= 390;++i)
                 copy_buffering_data[i]=_buffering[i];
              if (_scanf2(63,7,&_file) < 0) return 0;
              fp=fopen(_file,"w");
              for (i=0; i <= 390; ++i)
                 _buffering[i]=copy_buffering_data[i];
              for (i=0; ; ++i) {
                 gch=_buffering[i];
                 fputc(gch,fp);
                 if (gch == RET) break;
              }
              fclose(fp);
              _cls(49,7,78,18);
              _m_x=2; _m_y=2;
              return -2;
case _bs    : --_ma_x;
              if (_ma_x < _m_x) {
                 if (_ma_sto > 0) {
                    --_ma_y;
                    _ma_x=_m_x+42;
                 }
                 else
```

14

```
                 {
                    ++_ma_x;
                    return 0;
                 }
              }
              if (_ma_sto < 0) {
                 _ma_sto=0;
                 _ma_x=2;
                 return  0;
              }
              gotoxy(_ma_x,_ma_y);
              _mem_x=_ma_x;
              _mem_y=_ma_y;
              for (i=_ma_sto; i <= (_ma_sto - mag) ;++i) {
                 _buffering[i]=_buffering[i+1];
                 putchar(_buffering[i]);
                 next(0);
              }
              --_ma_sto;
              --_ma_x;
              if (_ma_x < _m_x) {
                 if (_ma_sto <= 0) {
                 _ma_sto=0;
                 _ma_x=_mem_x;
                 _ma_y=_mem_y;
                 gotoxy(_ma_x,_ma_y);
                 putchar(32);
                 return 0 ;
                 }
              _ma_x=_m_x+42;
              _ma_y=_mem_y;
              gotoxy(_ma_x,_ma_y);
              putchar(32);
              }
              gotoxy(_ma_x,_ma_y);
              putchar(32);
   not_space :
              _ma_x=_mem_x; _ma_y=_mem_y;
              gotoxy(_ma_x,_ma_y);
              return(0);
case  _del  : if (mag == 0) break;— 캐릭터 삭제
              _mem_x=_ma_x;
              _mem_y=_ma_y;
              gotoxy(_ma_x,_ma_y);
              for (i=(_ma_sto+1); i < (_ma_sto-mag); ++i) {
                 _buffering[i]=_buffering[i+1];
                 putchar(_buffering[i]);
                 next(0);
              }
              putchar(32); ++mag;
              _ma_x=_mem_x;
```

15

```
              _ma_y=_mem_y;
              gotoxy(_ma_x,_ma_y);
              return(0);
   case RET    : _mem_x=_ma_sto;     ┌ 계산식을 입력하고
                 i=_mem_x;           │      리턴을 친다.
                 _cls(48, 2,78, 3);  │ *주의 : 리턴 이후의
                                     └    문자는 지워진다.
                 for ( ; i <= 390; ++i)
                     _buffering[i]=RET;
                 _ma_sto=_mem_x;
                 gotoxy(1,10);
                 _in=1; _state_list(3);
                 _mem_x=_ma_x;
                 _mem_y=_ma_y;
                 for ( ; ; ) {
                     if (_ma_y == 18 && _ma_x == 43) break;
                     pokeb(BORD,_ma_x*2+_ma_y*160,0);
                     _next(0);
                 }
                 _ma_x=_mem_x;
                 _ma_y=_mem_y;
                 return 0;
   case _ins   : if (_in == 1) _in=0;— 입력키의 체크
                 else _in=1;
                 _state_list(3); —— insert 상태임을 표시
                 gotoxy(_ma_x,_ma_y);
                 return 0;
   default     : return(0);
  }
  return -1;
}

void _back() — insert 시 문자열을 뒤로 미는 프로시듀어
{
        int i;
        int _mem_x,_mem_y;
        _mem_x=_ma_x;
        _mem_y=_ma_y;
        if (mag == 0 || _ma_sto < 0) return ;
        for (i=(_ma_sto-mag); i >= _ma_sto ; --i)
           _buffering[i+1]=_buffering[i];
        for (i=_ma_sto+1 ; i <= (_ma_sto-mag) ; ++i) {
           putcc(_ma_x,_ma_y,_buffering[i]);
           next(0);
        }
        _ma_x=_mem_x;
        _ma_y=_mem_y;
      }

      int keyin(jump_condi)     모든 키의 입력을 받는 곳
      int jump_condi;
      {
```

```
   int ccc,i,jump,condi;
   if (jump_condi == 0) {
      cls_work_sheet();
      _m_x=2; _m_y=2;
   }
   if (jump_condi == 3) goto ban;
   mag=0; _ma_sto=-1;
init:
   for (i=0 ; i <= 389;++i) _buffering[i]=RET;
   _ma_sto=-1;
   _glo_ans = t = 0.0;
   _gl_err=0;
repeat:
   _ma_x=_m_x;
   _ma_y=_m_y;
ban:
   gotoxy(_ma_x,_ma_y);
   ccc=getkey();
   if ((ccc >= 256 || ccc < 32) && ccc != RET) {
      if (ccc == _ESC)
         if (jump_condi != 0) return -1;
      condi=_control_cha(0,ccc);
      if (ccc == _f5) return -1;
      if (condi == -2) {
         cls_work_sheet();
         goto init;
      }
      gotoxy(_ma_x,_ma_y);
      if (condi == -3) goto init;
      if (jump_condi == 2) return -1;
      goto ban;
   }
   ++_ma_sto;
   if (_ma_sto >= 390) {
      _ma_sto=-1;
      putchar(7);
      mag=-390;
      goto repeat;
   }
   ccc=tolower(ccc);
   if (ccc > 32 && ccc <= 'z' && jump_condi != 2) {
      putcc(_ma_x,_ma_y,ccc); next(jump_condi);
      if (_in == 0) _back();
      _buffering[_ma_sto]=ccc;
      goto ban;
   }
   if (ccc == RET) {
      if (_ma_sto == 0) goto init;
      if (jump_condi != 0) return 0;
      else {
```

16

```
            _control_cha(0,RET);
            condi=_interpreter(0);
            if (condi < 0) {
               mag=_ma_sto*-1;
               _ma_sto=-1;
               _gl_err=0;
               _glo_ans = t = 0.;
               goto repeat;
            }
            else
            {
               _m_x=2;
               _m_y=_ma_y;
               mag=0;
               if (_m_y > 18) {
                  _scroll_up();
                  _m_y=18;
               }
            }
            goto init;
         }
   }
   goto ban;
}
```

```
int _interpreter(jum)   { 입력된 수식을 해석한다.
int jum;                { 에러 발생시 -1을 리턴한다.
{
  int k,j,set,memi,ab,minus_fl;
  stack=0; dep=0;
  c1=0.0; c3=0; set=1;
  stackli[0].li=0;stackli[0].fn=0;

  t=0.0;
  glo_i=0;
  set=1;
 juml:
  minus_fl=1;
  ab=_buffering[glo_i];
  switch (ab) {
        case ','  :
        case RET  : push(t,_buffering[glo_i],dep);
                    if (set == 1) {
                       if (jum == 0) {
                          _error_print(8);
                          return -1;
                       }
                       else return -2;
                    }
                    if (dep != 0) {
                       if (jum == 0) _error_print(1);
```

17

```
                   return -2;
                }
                if (stack > 1) _stackcal();
                if (_gl_err != 0) return -1;
                pop();
                t=s1;
                if (jum == 0) {
                   if ((_ma_y + 1) > 18) {
                      _scroll_up();
                      _ma_y=17;
                   }
                   _dec(t,44);
                   _ma_y += 2;
                }
                _glo_ans=t;
                return 0;
case '('  : listfun[dep]=0;
            ++dep;
            ++glo_i;
            set=1;
            goto juml;
case ')'  : if (set == 1) {
               _error_print(8);
               return -2;
            }
            push(t,RET,dep);
            --dep;
            if (stack > 1) _stackcal();
            if (_gl_err != 0) return -1;
            pop();
            t=s1;
            if (listfun[dep] != 0)
               t=_cal_fun((listfun[dep]),t);
            set=0; ++glo_i;
            goto juml;
case ' '  : load[_buffering[glo_i+6] - 'a'] = t;
            return 0;
case '*'  :
case '/'  :
case '%'  :
case '+'  :
case 'P'  : ———— 'p'는 permutation
case 'C'  : memi=glo_i; ++memi; —— 'C'는 combination
           goto jump_li;
case '-'  : if (set == 1) {
               if (_tell(_buffering[glo_i+1]) != 0) {
                  _error_print(8);
                  return -1;
               }
               if (_buffering[glo_i+1] >= '0' &&
```

18

/PANTA/

19

```
            _buffering[glo_i+1] <= '9') {
            t=_atof();
            if (_gl_err != 0) return -1;
            ++glo_i;
        }
        else
        {
            if (_buffering[glo_i+2] >= 'a' &&
                _buffering[glo_i+2] <= 'z') {
                minus_fl=-1; ++glo_i;
                ab=_buffering[glo_i];
                goto jump_la;
            }
            else t=-1*load[glo_i+1-'a'];
        }
        set=0;
        goto juml;
        }
        memi=glo_i; ++memi;
        goto jump_li;
case SQR :          ──── 'sqr'는 squre를 의미
case '!' : _glo_ans=t; ──── '!'은 팩토리얼을 의미
        if (ab == SQR) t=four_operation(t,MUL,_glo_ans);
        else t=_n_fac(t);
        ++glo_i;
        set=0;        goto juml;

case 'Y' : if (set == 1) {
            _error_print(8);
            return -2;
        }
        if (_buffering[glo_i+1] != NSQ) {
            _error_print(3);
            return -2;
        }
        push(t,YYY,dep);
        if (stack > 1) _stackcal();
        if (_gl_err != 0) return -1;
        glo_i += 2;
        set=1;
        goto juml;

case POL : memi=(glo_i + 4); ──── 'pol'은 polynomial을 의미
        if (_buffering[glo_i+1] == 'p')
            if (_buffering[glo_i+2] == 'o')
                if (_buffering[glo_i+3] == 'l') goto jump_li;
        else
        {
            _error_print(3);
            return -2;
```

20

```
        }
case NSQ : if (set == 1) _error_print(8);─┤ 'nsq'는 n root를 의미
        push(t,NSQ,dep);
        set=1; glo_i += 2;
        goto juml;

case SQRT: if (set == 0) _error_print(8);─┤ 'sqrt'는 squre root를 의미
        listfun[dep]=SQRT;
        set=1;
        ++dep; glo_i += 2;
        goto juml;

jump_li : if (_buffering[glo_i] == RET) {
            _error_print(4);
            return -1;
        }
        push(t,ab,dep);
        if (stack > 1) _stackcal();
        if (_gl_err != 0) return -1;
        set=1;
        glo_i=memi;
        goto juml;
case '.' :
default :

jump_la :
        if (ab >= 'a' && ab <= 'z') {
            if (set == 0) _error_print(8);
            if (_buffering[glo_i+1] >= 'a' &&
                _buffering[glo_i+1] <= 'z') {
                j=searfun();
                if (j < 0) {
                    _error_print(3);
                    return -1;
                }
                else
                {
                    listfun[dep]=j*minus_fl;
                    minus_fl=1;
                    ++dep;
                }
                set=1;
            }
            else
            {
                set=0;
                if (jum == 2) t=glo_double;
                else t=load[_buffering[glo_i]-'a'];
            }
            ++glo_i;
            goto juml;
```

21

```
                }                       [glo_i] >= '0'
                if (_buffering[glo_i] <= '9' && _buffering
                    && set == 1) {
                    t=_atof();
                    if (_gl_err != 0) return -1;
                    set=0;
                    ++glo_i;
                    goto juml;
                }
        _error_print(8);
    }
    goto juml;
}

int _leng(ar) ── 자리수 출력
double ar;
{
    int i;
    char sstr[30];
    gcvt(ar,30,sstr); ── 실수를 문자열로 바꾼다.
    for (i=0; (sstr[i] >= '0' && sstr[i] <= '9') && sstr[i]
                                         != '.' ; ++i);
    if (sstr[i] == 'e') {
        sstr[0] = sstr[++i];
        sstr[1] = sstr[++i];
        sstr[2] = sstr[++i];
        sstr[3] = '\0';
        return atoi(sstr);
    }
    return i;
}

void pop() ── 숫자와 연산기호를 배열에서 꺼낸다.
{
    int i;
    if (stack <= 0) return;
    s1=stackli[0].li;
    s2=stackli[0].ch;
    s3=stackli[0].fn;
    for (i=1; i <= stack ; ++i) {
        stackli[i-1].li=stackli[i].li;
        stackli[i-1].ch=stackli[i].ch;
        stackli[i-1].fn=stackli[i].fn;
    }
    --stack;
}
          수   연산기호
void push(fi,se,th)   [수, 연산기호, depth를 배열에 집어 넣는다.
                depth
double fi;
int th;
char se;
{
```

```
    int i;
    if (_tell(se) == 0) {
        _error_print(8);
        return ;
    }
    for (i=stack ; i >= 0; --i) {
        stackli[i+1].li=stackli[i].li;
        stackli[i+1].ch=stackli[i].ch;
        stackli[i+1].fn=stackli[i].fn;
    }
    ++stack;
    stackli[0].li=fi;
    stackli[0].ch=se;
    stackli[0].fn=th;
}

int _tell(tel) —— 연산자의 우선순위를 출력
int tel;
{
  switch (tel) {
        case SQR:
        case '!':  return (1);
        case 'Y':
        case POL:
        case 'C':
        case 'P':
        case NSQ:  return (2);
        case '*':
        case '/':
        case '%':  return (3);
        case '+':
        case '-':  return (4);
        case ')':  return (5);
        case RET:  return (6);
  }
  return (0);
}

int searfun()  [ 입력된 함수를 token과 비교
{                틀리면 -1을, token에 존재하면
                 해당 수치를 리턴한다.
    char cop[5];
    int k,j;
    strcpy(&cop,"     ");
    k=0;
    for ( ; _buffering[glo_i] != '(' && k < 6; ) {
        cop[k] = _buffering[glo_i];
        ++glo_i; ++k;
    }
    if (k == 6) return -1;
    cop[k]=_buffering[glo_i];
```

22

```
    cop[++k]='\0';
    for (k=2 ; k <= 18 ; ++k)
        if (chrcmp(cop,_str1[k]) == 0) return(k);
    _error_print(8);
    return (-1);
}

int chrcmp(fi,se) —— 개개의 스트링과 비교
char fi[],se[];
{
   int k;
   for (k=0; fi[k] != '(' ; ++k)
       if ((fi[k] - se[k]) != 0) return(-1);
   return(0);
}

int _stackcal() —— 스택에 있는 것을 꺼내서 계산한다.
{
   int j;
   for (j=1; j <= 2;++j) {
       pop();
       hh=s1;
       jj=s2;
       kk=s3;
       pop();
       if (_tell(jj) >= _tell(s2) && (s3 == kk)) {
          t=_s_calcul(s1,s2,hh);
          push(t,jj,s3);
          if (stack == 1) {
             if (_gl_err != 0) return -1;
             else return 0;
          }
       }
       else
       {
          push(s1,s2,s3);
          push(hh,jj,kk);
          return 0;
       }
   }
   return -1;
}

double _atof() —— 문자열을 실수화한다.
{
    char cop[50],ch;
    int k,flag=1;
    if ((ch=_buffering[glo_i]) < '0') {
        if (ch == '-') flag=-1;   —— '-'의 부호 검사
           else if (ch == '+') flag=1;
                 else
```

23

```
                {
                    if (ch == '.') goto jum;
                    _error_print(4);
                    return(-1);
                }
        ++glo_i;
    }
jum:
   k=-1;
   --glo_i;
rep:
   ++glo_i;
   cop[++k]=_buffering[glo_i];
   if (k > 23) {
      _error_print(10);
      return -1;
   }
   if ((ch=_buffering[glo_i+1]) <= '9' && ch >= '0') goto rep;
   if (ch == '.') goto rep;
   if (ch == 'e' ) goto rep;
   cop[k+1]='\0';
   return (atof(cop)*flag);
}

double _s_calcul(__fi,__co,__se) ←— 스택에서 꺼낸 부분을 계산한다.
double __fi,__se;
char __co;
{
   int _l_fi,_l_se,con_in;

   _l_fi=_leng(__fi);
   _l_se=_leng(__se);

   con_in=__co;
   if ((_l_fi + _l_se) >= 90) goto _error_pr;
   switch (con_in) {
          case '+' :  return(__fi+__se);
          case '-' :  return(__fi-__se);
          case '*' :  if ((_l_fi * _l_se) >= 99) goto _error_pr;
                      return (__fi*__se);
          case '/' :  if (__se == 0.0) {
                         _error_print(9);
                         _gl_err=-1;
                         return 0;
                      }
                      return (__fi/__se);
          case 'Y' :  return (_n_squre(__fi,__se));←— 누승
          case POL :  return (_n_pol(__fi,__se)); ←— polynomial
          case NSQ :  return (_n_root(__se,__fi));
          case 'C' :  return (_n_c_fac(__fi,__se));←— combination
          case 'P' :  return (_n_p_fac(__fi,__se));←— permutation
```

24

```
        case '%' :  return (_mod(__fi,__se));
        default  :  _error_print(8);  ┤ 존재하지 않는 경우
                    return 0;         │ 에러를 표시하고
                                      └ 0 값을 리턴한다.
    }
_error_pr:
    _error_print(10);
    _gl_err=-1;
    return -1;
}

void _state_list(state)— 계산기의 상태를 표시
int state;
{
    switch (state) {
        case 1 :  video_string(50,22,ch_state[_deg]);
                  if (_deg == 0) _deg=1;  deg와 rad 표시
                  else _deg=0;
                  break;            caps key를 체크
        case 2 :  cps=bioskey(2);         _state[2]);
                  if (cps & 0x40) video_string(54,22,ch
                  else video_string(54,22,"     ");
                  break;
        case 3 :  video_string(59,22,"      ");
                  video_string(59,22,ch_state[3+_in]);
                  break;
        case 4 :  video_string(71,22,"      ");
                  video_string(71,22,ch_state[5+_st]);
    }                                 계산기의 상태표시
    video_string(66,22,"mode:");
}

void putcc(x,y,c)
int x,y,c;
{
    pokeb(BORD,x*2+y*160,c);
}

void _box(x1,y1,x2,y2,_mes,len,w)
int x1,y1,x2,y2;
char _mes[];
int len,w;
{
    int i,j;
    for (i=x1+1 ; i <= x2 ; ++i) {
        putcc(i,y1,196+9*w);
        putcc(i,y2,196+9*w);
    }
    for (i=y1+1 ; i <= y2-1 ; ++i) {
        putcc(x1,i,179+7*w);
        putcc(x2,i,179+7*w);
```

25

```
    }
    putcc(x1,y1,218-17*w);
    putcc(x2,y1,191-4*w);
    putcc(x2,y2,217-29*w);
    putcc(x1,y2,192+8*w);
    gotoxy(x1,x2);
    if (_mes[0] == NULL) return ;
    _glo=len;
    _glo=(x2-x1)/2+x1-_glo/2;
    for (i=0; i < len; ++i) putcc(_glo+i,y1,_mes[i]);
}

int _era_box(x1,y1,x2,y2)—— box를 지운다.
int x1,x2,y1,y2;
{
    int i;
    for (i=x1 ; i <= x2  ; ++i) {
        putcc(i,y1,32);
        putcc(i,y2,32);
    }
    for (i=(y1+1); i <= (y2-1) ; ++i) {
        putcc(x1,i,32);
        putcc(x2,i,32);
    }
}

void  _cls (xx1,yy1,xx2,yy2) ┤ (x, y) 좌표에 따라
int xx1,xx2,yy1,yy2;         └ 화면을 지운다.
{
    int i , j;
    for (i = xx1; i <= xx2 ; ++i)
        for (j = yy1 ; j <= yy2 ;++j)
            pokeb(BORD,i*2+j*160,0);
}
void _scroll_up()    ┌ rombios를 이용하지 않고
{                    │ video ram를 직접 이용하여
                     └ scroll up한다.
    int i,j,k;
    for ( j = 2 ; j <= 17 ; ++j)
        for( i = 2 ; i <= 44 ; ++i)
        {
            k=peekb(BORD,i*2+(j+1)*160);
            pokeb(BORD,i*2+j*160,k);
        }
    for (i=2; i <= 44;++i)
        pokeb(BORD,i*2+18*160,0);
}

void _item_pro(j) — 메뉴부분의 문자를 표시한다.
int j;
{
```

26

```
    int i,k=-1,l;
    if (j == 1) goto sub_pra;
    for (l=1; l <= 6;++l)
        for (i=1; i <= 5; ++i)
            video_string(49+6*(i-1),8+2*(l-1),_str1[++k]);
    return;
sub_pra:
    for (l=1; l <= 4;++l)
        for (i=1; i <= 3; ++i)
            video_string(49+9*(i-1),8+2*(l-1),_str2[++k]);
}

void insert_print(ch)  ┤ '_w_s_pr'에서 표시되는
char ch;               └ 문자를 insert 또는 over로 결정
{
  if (_in == 0) _back();
  _buffering[_ma_sto]=ch;
  putcc(_ma_x,_ma_y,ch);
}

void _w_s_pr(__fun,la) — 메뉴부분에서 선택한 부분을 워크시트에 표시
int __fun,la;
{
   int j;
   char dat[19]={"0000000000000000000"};
   gotoxy(_ma_x,_ma_y);
   --la;
   ++_ma_sto;
   switch (__fun) {
        case  1 : insert_print(YYY);
                  next(0);
                  ++_ma_sto;
                  insert_print(252);
                  break;
        case 11 : insert_print(253);
                  break;
        case 16 : insert_print('C');
                  break;
        case 17 : insert_print('P');
                  break;
        case 22 : gcvt(M_PI,19,dat);
                  for (j=0; j <= 15 ; ++j) {
                      insert_print(dat[j]);
                      ++_ma_sto;
                      next(0);
                  }
                  --_ma_sto;
                  return;
        case 21 : insert_print('1');
                  next(0);
                  ++_ma_sto;
```

27

```
                  insert_print('/');
                  break;
        case 20 : insert_print('!');
                  break;
        default : gotoxy(_ma_x,_ma_y);  gch != '\0';++j) {
                  for(j=0;(gch=_str1[__fun][j+1])!= ' ' &&
                     insert_print(_str1[__fun][j]);
                     ++_ma_sto;
                     next(0);
                  }
                  insert_print(_str1[__fun][j]);
                  break;
    }
    next(0);
}

void _error_print(_code) —— 이 프로그램에 쓰인 에러 리스트
{
 int i;
 char *error_list[]={
     "Tri function overflow",   "check '(' ')'",
     "invalid constant",        "invalid variable",
     "illegal decimal",         "does not convert to sin",
     "does not convert to cos", "does not convert to tan",
     "check '+' '-' '*' '\' '%'","division by zero",
     "decimal over flow",       "illegal decimal in n squre",
     "does not convert to bin", "does not convert to hex",
     "does not convert to dec", "does not convert to oct",
     "range of factorial is 90", "command.com does not find",
     "file does not exist",     "illegal pathname"};

   _gl_err=1; ——에러의 세트 여부
   if (_st != 0) return ;
   _cls(49, 2,78, 3);
   video_string(49,3,error_list[_code]);
}

void _dir() —— dir 리스트를 보여준다.
{
   char path[30],cb;
   char _dirli[100][13];
   int i,j=0,k,l,resu;
   struct ffblk ffbl; /* ffblk predefined in header 'dir.h' */

   cls_work_sheet();
   for (i=0; i <= 99; ++i)
      _dirli[i][0]='\0';
   i=0;
   video_string(2,2,"Dir mask:");
   if (_scanf2(11,2,&path) < 0) return ;
   resu=findfirst(path,&ffbl,0);
```

28

```
   strcpy(&_dirli[0],ffbl.ff_name);
   while (!resu) {
      ++i;
      resu=findnext(&ffbl);
      strcpy(_dirli[i],ffbl.ff_name);
   }
   if (i == 0) {
      _error_print(19);
      _gl_err=0;
      return ;
   }
   for ( ; j < i ; ) {
      for (l=0; l <= 15; ++l) {
         for (k=0; k <= 2; ++k) {
            if ((gch=_dirli[j][0]) >= 'A' && gch <= 'Z')
               video_string(2+k*14,l+3,_dirli[j]);
            else goto jum;
            ++j;
         }
      }
jum:
      video_string(2,18,"Next 'space'");
ban:
      gch=getkey();
      if (gch == _ESC) return ; — <esc>로 복귀
      if (gch != ' ') goto ban;
      cls_work_sheet();
   }
}

int _scanf2(x,y,_path) — 문자열을 입력한다.
int x,y;                   <esc>로 복귀
char _path[];
{
   int j,i;
   _m_x=x;
   _m_y=y;
   if (keyin(1) < 0) return -1;
   for (i=0; _buffering[i] != RET; ++i) ;
   _buffering[i]='\0';
   for (j=0; j <= i ; ++j) _path[j]=_buffering[j];
   return 0;
}

int _scanfd(x,y,sc_double) —┤ 숫자를 입력한다.
double *sc_double;           <esc>로 복귀
int x,y;
{
   _m_x=x; _m_y=y;
   if (keyin(1) < 0) return -1;
```

29

```
   if (_interpreter(1) < 0) return -1;
   if (_gl_err != 0) return -1;
   if (_buffering[0] == RET) return -1;
   *sc_double=t;
   return 0;
}

double _a_log(double,int),_exp();
double _in_squre(double,double);

int ask_end() ┤ 끝낼것인가를 묻는다.
{               끝이면 −1, 아니면 1을 리턴
   video_string(2,7,"Continue ?(y/n)");
   gotoxy(17,7);
   if (getkey() == 'y') return 1;
   return -1;
}
/* Pantagon calculator math lib */

int _sigma() — 기초통계 부분.
{
   int i,k;
   double s_dat[256];
   double def,sum,ave,sum2;
start:
   cls_work_sheet();
   i = k = 1;
   for ( ; ; ) {
      gotoxy(2,1+k);
      printf("Data(%3d)=",i);
      if (_scanfd(12,1+k,&s_dat[i-1]) < 0) goto cal;
      ++i;
      ++k;
      if (k > 17) {
         k=17;
         _scroll_up();
      }
   }
cal:
    if (i == 1) return -1;
    _cls( 2, 2,44,18);
    sum=0.;
    for (k=1; k < i; ++k)
       sum=_s_calcul(sum,ADD,s_dat[k-1]);
    if (_gl_err == 1) return -1;
    sum2=0.;
    for (k=1; k < i; ++k)
       sum2=four_operation(sum2,ADD,
          _s_calcul(s_dat[k-1],MUL,s_dat[k-1]));
    if (_gl_err == 1) return 0;
```

30

31

```
   ave=sum/(i-1);
   def=four_operation(sum2,DIV,i-1)-ave*ave;
   def=_n_root(def,2);
   _ma_y=1;
   video_string(2,2,"deflection :"); _dec(def,44);
   _ma_y=2;
   video_string(2,3,"average    :"); _dec(ave,44);
   _ma_y=3;
   gotoxy(2,4); printf("%cX        :",228); _dec(sum,44);
   _ma_y=4;
   video_string(2,5," X         :"); _dec(sum2,44);
   putcc(2,5,228); putcc(4,5,253);
   _ma_y=5;
   video_string(2,6,"item       :"); _dec(i-1,44);
   if (ask_end() > 0) goto start;
   return 0;
}

int display_Formula(i) — 화면에 'Formula f(a)'를 표시
int i;
{
   video_string(2,2,"Formula f(a)=");
   _m_x=16;
   _m_y=2;
   if (keyin(i) < 0) return -1;
   return 0;
}

void integral() —— 적분 부분
{
   int set,i;
   double mem1_x,mem2_x,area,cop1,cop2;
   double x1,x2;
start:
   cls_work_sheet();
   _ma_sto=-1;
   if (display_Formula(1) < 0) return;
   strcpy(_area_buff,_buffering);
   video_string(2,3,"start x:");
   if (_scanfd(10,3,&x1) < 0) return ;
   video_string(2,4,"end   x:");
   if (_scanfd(10,4,&x2) < 0) return ;
   if ((fabs(x1-x2)) < 0.0001) {      [ 시작과 끝의 좌표 차이가 0.0001일 경우 종료
      _error_print(10);
      return ;
   }
   area=0.0;
   strcpy(_buffering,_area_buff);
   video_string(2,5,"wait moment!!");
   for ( ; x1 <= x2 ; ) {
      glo_double=x1;
```

32

```
      if (_interpreter(2) < 0) break;   [ 해석중에 overflow나 문법에러 발생시 실행을 중단한다.
      mem1_x=_glo_ans;
      x1 += 0.01;
      glo_double=x1;
      if (_interpreter(2) < 0) break;
      mem2_x=_glo_ans;
      cop1=((mem1_x-mem2_x) < 0) ? mem1_x : mem2_x;
      cop1=cop1*0.01;
      cop2=0.01*fabs(mem1_x-mem2_x)/2;
      area = area + cop1+cop2;
   }
   _ma_y=5;
   video_string(2,6,"solution =>");
   if (_gl_err == 0) _dec(area,44);
   else video_string(40,6,"****");
end_position:
   if (ask_end() > 0) goto start;
   _area_buff[0]='\0';
   for (i=0; i <= 25; ++i)
      load[i]=0.0;
}

double _a_log(_fi,sel)              ] 이 프로그램의 라이브러리 부분
double _fi;
int sel;
{
   if (_fi > 9.9e9) {
      _error_print(10);
      return -1;
   }
   if (_fi < 0.0) {
      _error_print(4);
      return -1;
   }
   if (sel == 0) return(log10(_fi));
   else return(log(_fi));
}

double _squre (_fi)
double _fi;
{
   if (_fi > 3.0e9) {
      _error_print(10);
      return -1;
   }
   _glo_ans=_fi;
   return (_glo_ans*_fi);
}

double _n_pol(_fi,_se)
```

33

```
double _fi,_se;
{
   double res;

   if ((_fi=_squre(_fi)) < 0) {
      _error_print(10);
      return -1;
   }
   if ((_se = _squre(_se)) < 0) {
      _error_print(10);
      return(-1);
   }
   res = four_operation(_fi,ADD,_se);
   if (res < 0.0) {
      _error_print(10);
      return(-1);
   }
   if ((_se=_a_log(res,1)) < 0) {
      _error_print(10);
      return(-1);
   }
   if ((res = _exp(_se*0.5)) < 0) {
      _error_print(10);
      return(-1);
   }
   return res;
}

double _n_root(_fi,_se)
double _fi,_se;
{
   return _n_squre(_fi,1./_se);
}

double _n_squre(_fi,_se)
double _fi,_se;
{
   double _res,_res2;
   int _fi_leng,_se_leng;
   if (fmod(_se,1.) == 0.0) return _in_squre(_fi,_se);
   if (_fi < 0.0) {
      _error_print(11);
      return(-1);
   }

   _fi_leng=_leng(_fi);
   _se_leng=_leng(_se);

   if (pow((double)_se_leng,(double)_fi_leng) > 99.) {
      _error_print(10);
```

```
   return -1;
}
   return pow(_fi,_se);
}

double _in_squre(__fi,__se)
double __fi,__se;
{
  int _in_se,i;
  double res;
  _in_se=(int) __se;
  res=1;
  for (i = 1; i <= _in_se; ++i) {
     res=four_operation(res,MUL,__fi);
     if (_gl_err != 0) return -1;
  }
  return res;
}

double _n_fac(ij)
double ij;
{
   int _in_ij,i;
   double FAC;
   if (fmod(ij,1) < 0.0) {
      _error_print(4);
      return(-1);
   }
   if (ij > 90.) {
      _error_print(16);
      return(-1);
   }
   FAC=1;
   _in_ij=(int) ij;
   for (i=1; i <= _in_ij; ++i)
      FAC=FAC*(double) i;
   return FAC;
}

double _n_p_fac(_fi,_se)
double _fi,_se;
{
   double _res;
   if ((_se=(_fi-_se)) < 0.0) {
      _error_print(4);
      return(-1);
   }
   _res= _n_fac (_fi)/_n_fac(_se);
   if (_res < 0.0) return(-1);
   return _res;
```

34

```
}

double _n_c_fac(_finum,_senum)
double _finum,_senum;
{
   double fi,se,_res;

   if ((_res=(_finum-_senum)) < 0.0) {
      _error_print(4);
      return(-1);
   }
   if (_res == 0) _res=1;
   fi=_n_fac(_senum);
   se=_n_fac(_res);
   _res=_n_fac(_finum)/four_operation(fi,MUL,se);

   if (_res < 0.0) {
      _error_print(10);
      return(-1);
   }

   return _res;
}

double _exp(_fi)
double _fi;
{
   if (_fi > 150.) {
      _error_print(10);
      return(0);
   }
   return exp(_fi);
}

double _tri_cal(_fun,bun,_fi)
int _fun,bun;
double _fi;
{
   switch (_fun) {
        case 1  :  if (_deg == 0) _fi=_fi*M_PI/180;
       deg계산      if (_fi >= 9.9e9 || _fi <= -9.9e9) {
                      _error_print(0);
                      return(0);
                   }
                   switch (bun) {
                          case 1 : return (sin(_fi));
                          case 2 : return (cos(_fi));
                          case 3 : return (tan(_fi));
       rad계산      }
        case 2  :  if (_fi > 1) {
```

35

```
                   _error_print(10);
                   return -1;
                }
               if (_fi <  -1) {
                  _error_print(10);
                  return -1;
               }
               switch (bun) {
                      case 1:t = asin(_fi);
                             break;
                      case 2:t = acos(_fi);
                             break;
                      case 3:t=  atan(_fi);
               }
               if (_deg == 0) t = t * convert_to_deg;
               return t;
        case 3 : if (_fi >= 9.9e1 || _fi <= -9.9e1) {
 쌍곡선함수 계산     _error_print(0);
                  return -1;
               }
               switch (bun) {
                      case 1 : return (sinh(_fi));
                      case 2 : return (cosh(_fi));
                      case 3 : return (tanh(_fi));
               }
}
   return (-1);
}

double _mod(_fi,_se)
double _fi,_se;
{
   if (fmod(_se,1) != 0.0) {
      _error_print(4);
      return -1;
   }
   return fmod(_fi,_se);
}

void _bin(item) — 2진수로 입력된 수를 변환
double item;
{
   int i,j;
   float res;

   if (item > 2.9e9 || item < 0.0 || fmod(item,1) < 0.0 ) {
      _error_print(12);
      return ;
   }
   for (i=43; i > 3; --i) {
       res=(float)fmod(item,2.0);
```

36

37

```
      j='0'+res;
      putcc(i,_ma_y,j);
      res = res / 2.0;
      if (item < 2.0) break;
      item = item / 2.0 - res;
   }
}

void _hex(item) —— 16진수로 변환
double item;
{
   char hex[16]={"0123456789ABCDEF"};
   int i;
   double res;
   if (item > 2.9e9 || item < 0.0 || fmod(item,1) < 0.0) {
      _error_print(12);
      return ;
   }

   for (i=43; i > 3; --i) {
      res=fmod(item,16.);
      putcc(i,_ma_y,hex[(int)res]);
      res = res / 16.;
      if (item < 16.) break;
      item = item / 16. - res;
   }
}

void _dec(_glo_ans,x) —— 화면에 계산결과를 표시
int x;
double _glo_ans;
{
   char dec_dat[30]={"000000000000000000000000000000"};
   int set=0,i,j;
   gcvt(_glo_ans,30,dec_dat);
   j=0;
   for (i=29; i >= 0; --i) {
      if (dec_dat[i] != '0') set=1;
      if (set == 1) {
         ++j;
         pokeb(BORD,(x-j)*2+(_ma_y+1)*160,dec_dat[i]);,
      }
   }
}

void _oct(_glo_ans) —— 8진수로 변환
double _glo_ans;
{
   int i;
   double res;
```

38

```
   if (_glo_ans > 2.9e9) {
      _error_print(12);
      return ;
   }
   for (i=43; i > 3; --i) {
      res=fmod(_glo_ans,8.);
      putcc(i,_ma_y,'0' + (int)res);
      res = res / 8.;
      if (_glo_ans < 8.) break;
      _glo_ans=_glo_ans / 8. - res;
   }
}

void _graph() —— 그래프를 그린다.
{
   int i,j,jo;
   double xi,yi,inc;
init:
   cls_work_sheet();
   _st=2;
   _state_list(4);
   if (display_Formula(1) < 0) return;
   else save_vi();
   video_string(2,2,"clear buffer ?(y/n)"); getkey();
   xi= -176./(20./_x_inter);
   inc=_x_inter/20.;
   if (BORD == 0xb000) _set_hur();
   else _set_cga();
   if (gch=='y') clear_graphic();
   for (i=0; i <= 352; ++i) {
      plot(i,0); plot(i,200); plot (i,100);
      if ((int)(i%20) == 0) for (j=95; j <= 105; ++j)
                                          plot(i-3,j); }
   for (i=0; i <= 200; ++i) {
      plot(0,i); plot(352,i); plot(176,i);
      if ((int)(i%20) == 0) for (j=172; j <= 181; ++j)
                                          plot(j,i); }
                                     화면에 좌표축 표시
   for (i=1; i < 352 ; ++i) {
      glo_double=xi*_x_inter;
      _interpreter(2);
      yi=_glo_ans*(20./_y_inter)*-1.;
      if (yi > 99) yi=99.;
      if (yi < -99 ) yi=-99.;
      xi += inc;
      if (_gl_err == 0) plot(i,(int)yi+100);
   }
   getkey(); —— 그래프를 그리고 키의 입력을 기다린다.
   _set_text();
   _cls(1,80,1,25);
   load_vi();
   _box(47, 6,79,20,"Menu"          , 4, 0);
```

39

```
   _cls(48,7,78,19);
   if (ask_end() > 0) goto init;
   for (i=0; i <= 25; ++i)
      load[i]=0.0;
}

double _cal_fun(sor,_t) —— 함수를 항목에 따라 계산한다.
int sor;
double _t;
{
   int ju,so;
   double flag;
   flag=1.;
   if (sor < 0) {
      flag=-1.;
      if (sor == SQRT) flag=1;
      else sor = abs(sor);
   }
   else
      if (sor == abs(SQRT)) {
         flag=-1.;
         sor=sor*-1.;
      }

   switch (sor) {
         case  6  : return (_exp(_t)*flag);
         case 10  : return (_a_log(_t,0)*flag);
         case 15  : return (_a_log(_t,1)*flag);
         case SQRT: return (_n_root(_t,2.)*flag);
   }
   ju=3;
   if (sor <= 4) ju = 1;
   if (sor > 5 && sor <= 9) ju=2;
   so=sor%5-1;
   glo_double=_tri_cal(ju,so,_t);
   return glo_double*flag;
}

double four_operation(_fi_num,_op,_se_num) —— 사칙연산을 계산
double _fi_num,_se_num;
char _op;
{
   int _leng_fi,_leng_se;
   char convert_to_char[27];
   gcvt(_fi_num,27,convert_to_char);
   _leng_fi=_leng(_fi_num);           ———— 자리수 계산
   gcvt(_se_num,27,convert_to_char);
   _leng_se=_leng(_se_num);

   if ((_leng_fi+_leng_se) >= 90) goto _error_pr
                                       자리수가
                                       넘치면
                                       에러표시
```

40

```
    switch (_op) {
          case '+' : return (_fi_num+_se_num);
          case '-' : return (_fi_num-_se_num);
          case '*' : if ((_leng_fi * _leng_se) >=90) goto _error_pr;
                     return (_fi_num*_se_num);
          case '/' : if (_se_num == 0.0) {
                          _error_print(9);
                          _gl_err=-1;
                          return 0;
                     }
                     return (_fi_num/_se_num);
          default  : _error_print(8);
                     return 0;
    }
  _error_pr:
    _error_print(10);
    _gl_err=-1;
    return 0.;
  }

  char __buff[2000];

  void save_vi(),load_vi();

  void save_vi()
  {
    int i,j,l;
    unsigned int k=0,cl;
    union REGS r;
    r.x.ax=0x0f00;
    int86(0x10,&r,&r);
    mode=r.x.ax;
    for (i=0; i <= 80; ++i)
        for (j=0; j <= 25; ++j) {
            cl=i*2+j*160;
            l=peekb(BORD,cl);
            pokeb(BORD,cl,0);
            __buff[k]=l;
            ++k;
        }
  }

  void load_vi()        ────── 화면에 그림을 표시한다.
  {
    int i,j,l;
    unsigned int k=0;
    union REGS r;
    r.x.ax=mode;
    int86(0x10,&r,&r);
    for (i=0; i <= 80; ++i)
```

41

```
        for (j=0; j <= 25 ; ++j) {
            l=__buff[k];
            pokeb(BORD,i*2+j*160,l);
            ++k;
        }
  }

  void _o_s_shell()      ────── shell 부분
  {
    char *path;
    int i,j,l;
    path=getenv("COMSPEC");
    save_vi();
    _set_text();
    if ((spawnl(0,path,path,NULL)) < 0) _error_print(17);
    load_vi();
  }

                 ────── 저장한 수식을 로드
  int _load()
  {
    int i;
    FILE *fp,*fopen();
  start:
    _cls(49, 7,78,18);
    video_string(49,7,"Load filename:");
    if ((_scanf2(63,7,&_file)) < 0) return -1;
    fp=fopen(_file,"r");
    if (fp == NULL) {
       _error_print(18);
       fclose(fp);
       goto start;
    }
    _cls(49, 7,78,18);
    cls_work_sheet();
    _ma_sto=0;
    for (i=0; i <= 390; ++i) _buffering[i]=RET;
    for (i=0; ; ++i,++_ma_sto) {
        gch=fgetc(fp);
        if (gch < 0) break;
        _buffering[i]=gch;
    }
    _ma_x=_m_x=2;
    _m_y=_ma_y=2;
    i=0;
    gotoxy(_ma_x,_ma_y);
    for ( ; i < _ma_sto ; ) {
        putchar(_buffering[i]);
        next(1);
        ++i;
    }
```

42

```
    return 0;
  }

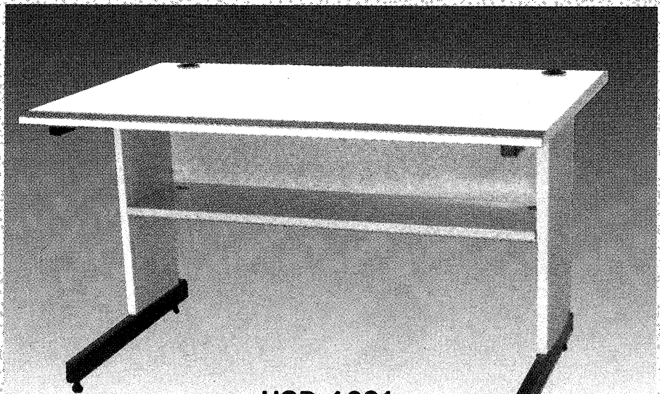
  void _run() ── area에 있는 수식을 해석 실행한다.
  {
    int jo;
    double a;
  start:
    cls_work_sheet();
    video_string(2,2,"Value 'a'?");
    _scanfd(12,2,&a);
    load[0]=a;
    if (_area_buff[0] == '\0') {
       cls_work_sheet();
       return;
    }
    strcpy(_buffering,_area_buff);
    _interpreter(1);
    video_string(2,4,"result =>");
    _ma_y=3;
    if (_gl_err != 0) video_string(40,4,"****");
    else _dec(_glo_ans,44);
    if (ask_end() > 0) goto start;
  }

  void area()      ────── area에 입력부분
  {
    int j;
  start:
    video_string(2,2,"Formula f(a)=");
    _ma_x=16; _ma_y=2; _m_x=16; _m_y=2;
    if (_area_buff[0] != '\0') {
       j=0;
    for ( ; _area_buff[j] != RET; ) {
        putcc(_ma_x,_ma_y,_area_buff[j]);
        ++j;
        next(0);
    }
    strcpy(_buffering,_area_buff);
    _ma_sto= (--j);
    mag=0;
    if (keyin(3) < 0) return ;
  }
  else if (keyin(3) < 0) return ;
  strcpy(_area_buff,_buffering);
  if (ask_end() > 0) goto start;
}
```

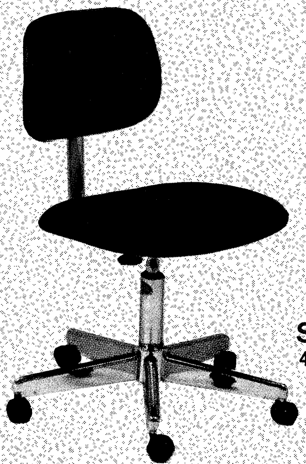
T·H·E·B·E·S·T·S·E·L·E·C·T·I·O·N

I.Q. MAN

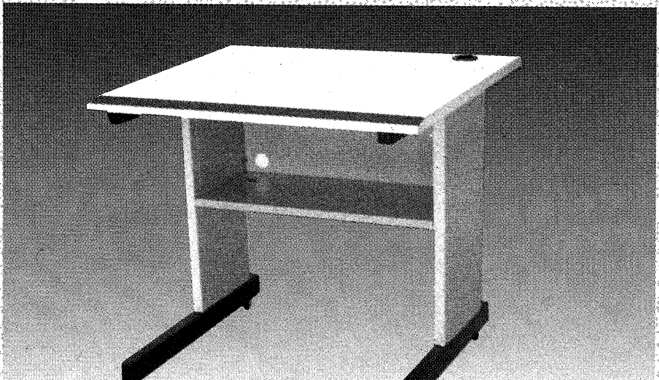
컴퓨터테이블



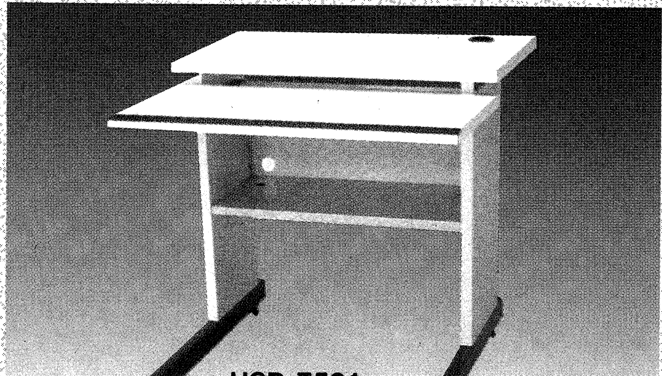
HSD 1201
1200(W)×700(D)×670(H)



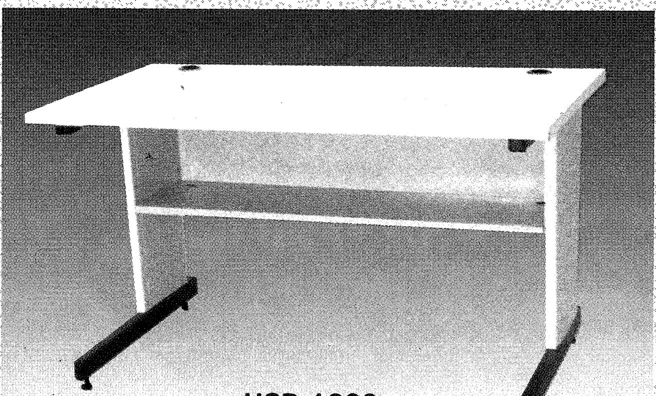
SK-8701-BS
450(W)×480(D)×750(H)



HSD 7501
750(W)×700(D)×670(H)



HSD 7521
750(W)× 상400(D) 하350(D) ×670(H)



HSD 1202
1200(W)×700(D)×670(H)



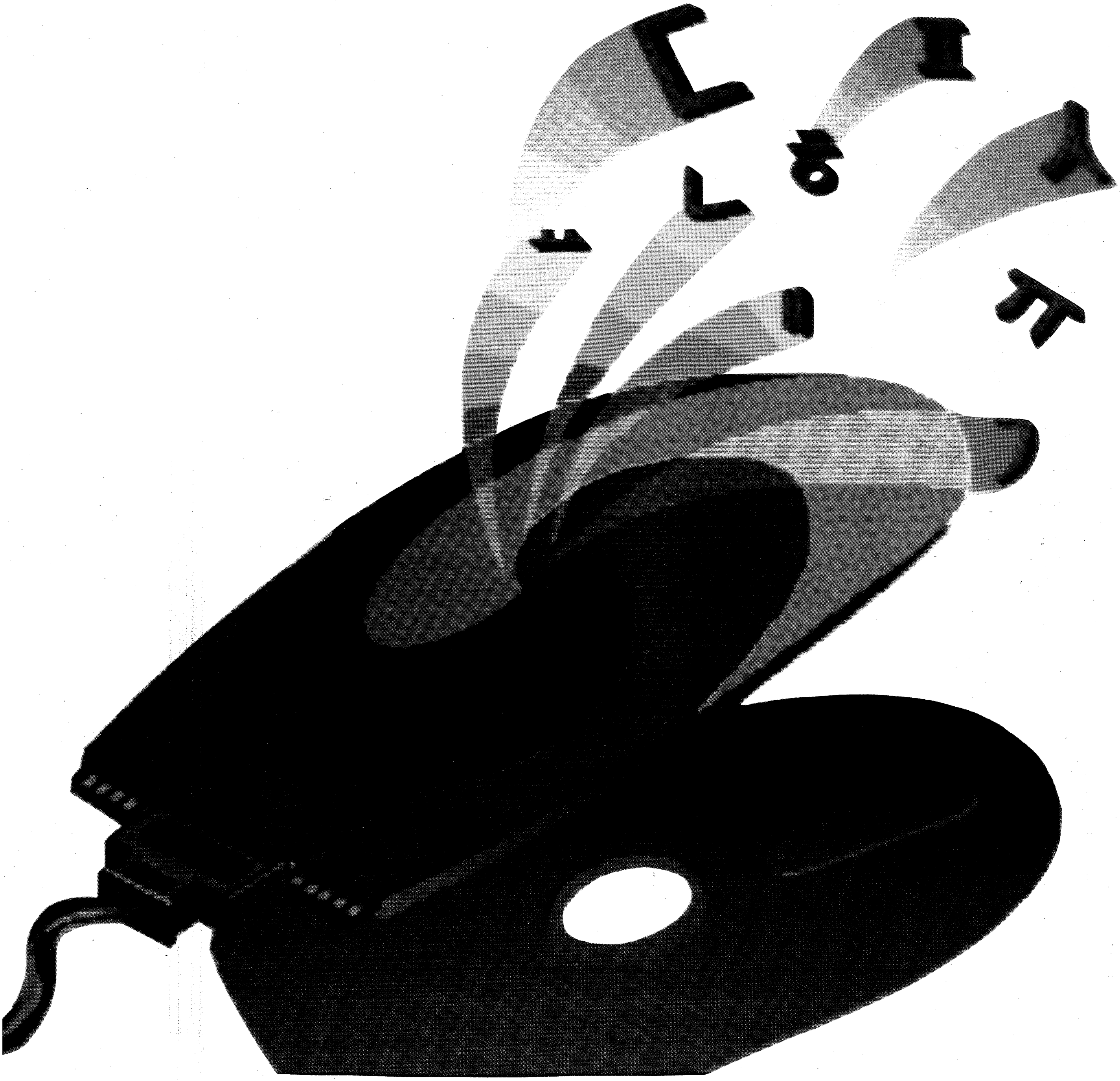
SK-8701-AG
560(W)×480(D)×750(H)

※ 전국지방대리점 및 영업사원 모집중

(株) 現 新 物 産

본사·공장 : 충남 천안시 두정동 52번지
TEL : (0417) 2－2114, FAX : (0417) 61－2168
판매장 : 서울특별시 용산구 원효로 3가 51-30 (원효전자상가 2층 111호)
TEL : (02) 713－5765～6, 717－6611 ㉑ 2120(代)

한글입출력에 관한



연구 ■특집

한글 입출력에 관한 연구

한글 코드 체계의 불합리한 점들은 비단 어제 오늘의 이야기가 아니라 오래전부터 지적되어 온 사항이다.
7비트 N바이트 코드, 7비트 3바이트 코드, 8비트 3바이트 코드, 2바이트 조합형 코드, 2바이트 완성형 코드 등 각 코드들 모두가 장·단점을 가지고 있으므로 어느 것이 좋고 나쁘다는 말을 하기는 어렵다. 그러나 코드 체계가 달라서 겪어야 하는 우리들의 불편은 이루 말할 수 없다.
따라서 여기에서는 이러한 여러 한글 코드들의 장·단점, 한글 입력자판 배열의 문제점과 그 개선책, 한글 입력 오토메타의 구성 원리에 대해서 알아봄으로써 한글 처리 문제에 대한 위상을 새로이 정립해 보고자 한다.

김한택/자유기고가

이글을 시작하면서

일본에서 우리 가요를 가라오케로 만들어 수출한다는 이야기는 벌써 기억에서 가물거릴 정도로 옛날 이야기가 되어가고 있다. 얼마전 KOEX에서 열린 제3회 한국 소프트웨어 공모전에 관람하러 갔다가 대만에서 들여온 한글카드를 보았을 때 왜 갑자기 그 생각이 되살아났을까? 당연히 우리가 해야 할 일이었음에도 불구하고 바다건너 사는 이들이, 그것도 한글을 사용하지도 않는 우리와는 다른 문화권의 사람들이 해냈다는데에 대해서 몹시도 자존심이 상했고 한편으로는 마치 내가 아끼는 물건을 잃어버린 듯한 아쉬움과 안타까움을 느껴야만 했다. 이 나라에는 인재가 없어서 못했을까 아니면 기술이 부족해서 못했을까? 그것도 아니면 무엇 때문일까?

컴퓨터가 한국인에 의해서 발명되지 못한 덕택에, 오늘날 우리는 컴퓨터를 생산하는 기업에서 사용자에 이르기까지 한글처리 문제 때문에 고심을 해야만 하는 입장에 놓여 있다. 새로운 버전의 소프트웨어가 발표되더라도 일단 한글을 사용할 수 있는지의 여부부터 확인해야 하고, 새로운 OS나 신기종의 컴퓨터가 발표되면 그것이 대형이든 개인용이든 제일 먼저 한글을 쓸 수 있도록 해야 하는 것이 우리 컴퓨터 업계의 현황이다. 따라서 한글처리 문제는 컴퓨터 회사만의 일도 아니요, 프로그래머만의 일도 아니요, 우리 모두의 문제이다.

끈래들어 필자 주변에는 한글 워드프로세서를 만드는 친구도 있고, 또 한글 통신 프로그램을 만드는 친구도 있다. 하지만 이들이 프로그래머로서 작업을 할 때에 한글처리에 관한 어떤 체계적인 지침서도 발견할 수 없었던 것이 우리의 현실이다. 자기가 애써 연구한 결과를 아무런 댓가없이 세상에 내놓는다는 것도 어려운 일이긴 하겠지만, 한글 입출력 처리와 같은 기초부분은 이제 더 이상 노하우가 아니라 모두가 함께 공유해야할 지식의 대상이 되어야할 것이다.

한글에 관련된 소프트웨어를 제작하려는 이마다 매번 한글 입출력 부분부터 새롭게 작성해야 한다면 이는 우리나라의 소프트웨어 발전에 있어서도 크나큰 손해이다. 여기에서 지식의 공개념이 필요하다. 기초가 되는 지식이나 기술은 어느 특정인의 소유 대상이 되어서는 안된다. 그런 것은 공개되어

야 하고 가급적 널리 알려져야만 더 큰 발전을 기대할 수 있다.

특히나 요즈음 국내의 컴퓨터 업계는 표준화를 지향하고 있는 추세이므로, 지금까지 난립되어 있는 한글처리 방식은 점차적으로 하나의 물줄기로 합류되어야 한다는 측면에서 한글의 입출력에 관해서는 비단 시스템 설계자나 프로그래머뿐만 아니라 매일같이 직접 몸으로 부대끼며 체험하는 사용자들도 함께 관심을 가져야 할 분야가 아닌가 생각한다.

이 글에서는 한글의 입출력은 어떻게 이루어지며 거기에서 고려해야 할 문제점들은 어떠한 것들이 있는지에 관하여 다루고자 한다. 한글처리에 관하여는 워낙 그 영역이 광범위하기 때문에 모든 것을 다 다루자면 수박 겉핥기가 되기 쉽고, 그렇다고 하나하나 너무 자세히 다루다 보면 지엽적으로 흐를 것 같아 가급적 중용을 기하면서 쓰도록 하겠으나, 형편없는 글재주가 앞서 가려하는 마음을 충족시켜줄 수 있을 지는 의문이다. 그리고 한글처리하면 모든 기종의 컴퓨터에 걸친 문제이긴 하지만 이 글의 성격상 우리 주위에 널리 보급되어 있는 16비트 개인용 컴퓨터를 위주로 다루게 되었음을 미리 밝혀둔다.

한글처리 수준에 따른 분류

한글을 처리하는데 있어서 어느 수준에서 처리할 것인지에 관한 문제이다. 한글을 사용하는 프로그램은 어느 것이나 다음에 열거된 사항 중 하나에 해당한다. 이해를 돕기위하여 요즈음 사용되고 있는 프로그램을 예로 들면서 설명하기로 한다.

(1) BIOS 수준에서의 처리

NKP나 한글도깨비와 같은 프로그램이 대표적인 소프트웨어에 해당한다. 독자적인 하나의 프로그램이므로 OS의 버전에 구속되지 않고 독립적으로 사용할 수 있는 잇점이 있다. 즉, DOS 버전이 3.1이든 3.3이든 구애됨이 없이 사용가능하다는 뜻이다. 주로 램상주 프로그램으로서 키보드 인터럽트(INT 16H)와 비디오 인터럽트(INT 10H) 핸들러를 통하여 한글 입출력 기능을 수행한다.

(2) OS수준에서의 처리

한국 마이크로소프트사의 한글 MS-DOS가 대표적인 경우에 해당하는데, 이는 OS 수준에서 한글을 처리하고 있기 때문에 앞에서 설명한 BIOS 수준에서 처리하고 있는 기본적인 한글 입출력외에도 한글 화일이름을 사용할 수 있으며 또한 메시지가 한글로 표시되기 때문에 사용자에게는 훨씬 친근감을 줄 수 있다.

그러나 한글을 처리하는 부분이 OS내에 포함되어 있기 때문에 독립된 프로그램이 아니므로 해당 DOS 버전에서만 사용해야 한다는 제약도 있다. 대개 hecon.sys와 같은 디바이스 드라이버(Device Driver) 의 형식으로 되어 있으며 이를 config.sys에서 device=hecon.sys와 같이 지정해 줌으로써 사용한다.

K한글의 경우에는 한글 화일이름을 사용할 수 있게 했다는 점에서 OS 수준에 접근하고 있다고 볼 수는 있지만 그것은 DOS의 일부를 수정하여 한글 화일이름을 사용할 수 있게 했을뿐, DOS 전체에 한글처리를 한 것이 아니며 그 형태도 디바이스 드라이버의 형식이 아니라 램상주 프로그램의 형식을 취하고 있으므로 BIOS 수준에서 한글처리를 하고 있다고 보는 편이 나을 것 같다.

(3) 응용프로그램에서의 처리

현재 대표적인 프로그램으로서는 이찬진씨의 한글 워드프로세서(일명 아래아 한글)가 있다. 이는 모든 한글 입출력을 NKP나 한글 MS-DOS와 같이 다른 외부적인 프로그램에 의존하지 않고 자체 프로그램 내부에서 처리하고 있다. 이 방법은 사용자에게 있어서는 가장 바람직하며 완벽한 한글처리가 가능하지만, 프로그램 제작자의 측면에서 보면 프로그램 내부에 자체적으로 한글을 처리해 주는 부분을 포함시켜야 하므로 시간과 노력이 훨씬 더 들뿐만 아니라 한글처리에 있어서도 통일성을 기대하기가 어렵다. 이를테면 한/영 변환키를 어느 것으로 할 것인가 하는가에 있어서도 프로그램 제작자의 취향에 따라 달라지게 된다.

한글코드

요즈음 컴퓨터를 이용한 통신이 발전하면서 국내에도 여러개의 BBS(Bulletin Board System)가 설치되어 운영되고 있다. 코드체계가 다른 것은 아예 언어가 다른 것과 마찬가지로 의사소통이 이루어지지 않으므로 통신에서만큼 한글코드의 심각성을 느끼는 분야도 드물다. 다행히도 리볼트나 메디콤과 같이 여러가지 코드체계를 사용할 수 있는 통신 프로그램들이 보급되어 그나마 우리는 통신의 혜택을 누리고 있는 지도 모른다.

한글코드 문제는 비단 통신분야뿐만 아니라 회사나 학교에서 작성한 데이터를 집에 와서 보려면 반드시 코드변환하는 작업을 거쳐야만 한다든지, 프로그램 하나 구입하려고 해도 가지고 있는 컴퓨

터 기종이 행망용인지 조합형 한글을 사용하는 것인지 확실히 알고 가야 헛걸음을 하지 않는다든지, 또는 프로그램을 제작하는 사람에게 있어서도 여러 기종에서 동작하는 프로그램을 만들려면 각기 다른 코드를 모두 지원해야 하는 부담을 겪어야 한다든지 하는 식으로 이미 우리 생활의 구석구석에서 알게 모르게 우리를 괴롭히고 있는 것이다.

국가에서 제정한 한글코드의 표준안이 없어서 이렇게 코드가 난립되어 있는 것은 아니다. 표준안이 마련되어 있다고 하더라도 그것은 강제적인 구속력을 가진 것이 아니라 어디까지나 권장안에 불과하므로 그것을 따라야 하는 의무는 없는데다가 표준안 자체가 합리적이지 못하고 많은 문제점을 가지고 있어서 업계나 사용자들의 호응을 얻지 못하였던 것이다. 그러던 터에 행정전산망에 KS C 5601 표준안이 채택됨으로써 비로소 표준코드가 우리 주위에 급속도로 세력을 뻗치기 시작하였다.

한글처리를 하는 소프트웨어를 제작하려면 반드시 코드 문제를 고려해야만 한다. 모두가 하나의 코드체계를 사용하고 있다면 이러한 문제는 없겠지만 지금처럼 여러가지 코드가 존재하는 한 최소한 조합형 코드와 완성형 코드가 어떠한 원리하에 만들어졌는지, 그리고 두가지 코드간의 장단점 정도는 이해하고 있어야 할 것이다.

한글은 天(초성), 地(중성), 人(종성)을 그 제자원리로 창제되었다. 아무리 완성형 코드가 국가 표준으로 채택되었다고 할지라도 한글의 제자원리를 바탕으로 조합형 코드가 만들어졌으므로 한글 입력 오토메타에 있어서나, 한글 포스트스크립, 한글 문자인식 등 여러분야에 있어서 조합형의 기본원리를 이용하여야만 훨씬 간편하게 처리할 수 있다. 이러한 의미에서 본고에서는 코드체계를 결코 빠뜨릴 수 없는 중요한 문제로 인식하고 여러가지 코드체계와 그 특성을 변천과정속에서 알아보고자 한다. 단, 흐름을 잃지 않기 위하여 요즈음 잘 사용하고 있지 않거나 그다지 중요성이 없는 코드체계에 관하여는 주마간산(走馬看山)식으로 다룰 수 밖에 없는 점을 이해해주기 바란다.

영어 알파벳이야 26자 밖에 안되므로 대·소문자 모두 합쳐보아야 52자, 거기에 자판에 있는 숫자와 부호 그리고 제어코드까지 모두 합쳐도 7비트만으로 얼마든지 구분하여 표시할 수 있다. 하지만 한글은 이보다는 훨씬 글자수가 많기 때문에 이를 표시하기 위해서는 좀 다른 방법을 강구해야 한다.

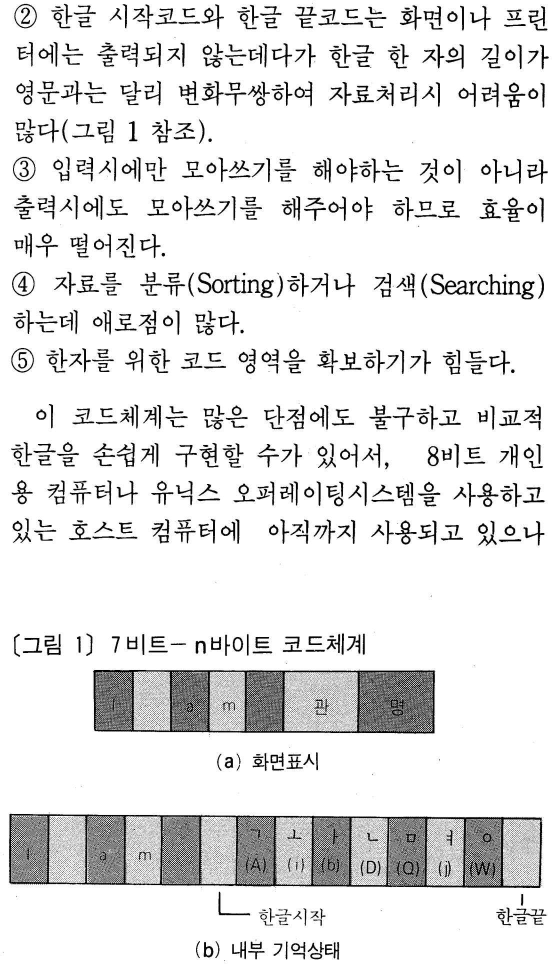
(1) 7비트-n바이트 코드체계

한글코드의 크기가 가변적이므로 흔히 n바이트 코드라고 부르는 코드체계이다. 영문 제어코드 중 2개를 그림 1(b)와 같이 한글 시작코드(SO)와 한글 끝코드(SI)로 지정한 다음, 한글시작코드가 나타난 이후의 코드들은 한글을 의미하며 한글 끝코드 이후의 코드들은 영문 코드를 의미하도록 만든 코드체계이다. 한글을 영문코드와 똑같이 7비트로 취급하기 때문에 7비트만 처리할 수 있는 컴퓨터나 OS(Operating System)에서 한글을 손쉽게 사용할 수 있는 장점도 있지만 다음과 같은 단점들이 있다.

① 한글과 영문이 같은 코드 영역을 사용하고 있으므로 프로그램에서 어느 한 문자에 대해 이것이 영문인지 한글인지 구별하기가 어렵다.
② 한글 시작코드와 한글 끝코드는 화면이나 프린터에는 출력되지 않는데다가 한글 한 자의 길이가 영문과는 달리 변화무쌍하여 자료처리시 어려움이 많다(그림 1 참조).
③ 입력시에만 모아쓰기를 해야하는 것이 아니라 출력시에도 모아쓰기를 해주어야 하므로 효율이 매우 떨어진다.
④ 자료를 분류(Sorting)하거나 검색(Searching)하는데 애로점이 많다.
⑤ 한자를 위한 코드 영역을 확보하기가 힘들다.

이 코드체계는 많은 단점에도 불구하고 비교적 한글을 손쉽게 구현할 수가 있어서, 8비트 개인용 컴퓨터나 유닉스 오퍼레이팅시스템을 사용하고 있는 호스트 컴퓨터에 아직까지 사용되고 있으나

〔그림 1〕 7비트－n바이트 코드체계



점차 그 사용이 줄어들고 있는 추세이다.

한글구현이 용이한 이유는, 우선 한글코드와 영문코드가 같은 코드영역을 사용하고 있으므로 OS는 거의 수정할 필요가 없는 대신 터미널에서만 한글을 지원하도록 하면 되기 때문이다. 따라서 적은 노력과 적은 비용으로 한글처리가 가능한 잇점이 있다.

개인용 컴퓨터로는 대표적으로 애플 컴퓨터의 CALL 3327 한글이 이 코드체계를 사용하고 있으며, 우리 주위에서는 현재 한국데이타통신의 한글 전자사서함과 천리안 서비스가 이 코드체계를 사용하고 있고, 삼보컴퓨터의 제닉스(Xenix) OS 역시 한글처리를 위해 이 코드체계를 따르고 있다.

(2) 7비트-3바이트 코드체계

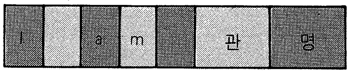
7비트-n바이트 코드체계의 한글 길이가 가변적이었던 것에 비해 **그림 2**와 같이 한글이 항상 초성, 중성, 종성의 3바이트로 고정됨으로써 자료의 분류 및 검색이 용이해졌으나 n바이트가 지닌 다른 문제점들에 대해서는 근본적으로 해결하지 못하고 있다. 현재 이 코드체계는 거의 사용되지 않고 있다.

(3) 8비트-3바이트 코드체계

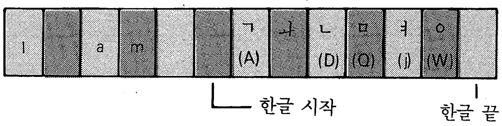
7비트 코드체계는 영문코드의 영역을 그대로 한글코드에 할당하여 사용하는 대신 한글 시작코드와 한글 끝코드를 사용하여 한글과 영문을 구별하는 기준으로 삼았다. 그로 인해 하나의 코드만 보아서는 그것이 한글인지 영문인지 알 수가 없고, 내부에 기억하고 있는 코드를 처음부터 해당 글자의 위치까지 검색하여 한글 시작코드와 한글 끝코드가 어떻게 들어있는지 추적해야만 했다.

이와 같은 문제점을 해결하기 위한 방안으로서 8비트 코드체계가 생겨나게 되었는데, 이는 한글코드를 영문코드와는 별도로 8비트 영역(MSB가 1인 코드 영역, 즉 코드값이 128 이상인 영역)내에 할당함으로써 하나의 코드를 놓고 볼 때에 바로 한글인지 영문인지 식별할 수 있게 하였다(**그림 3** 참조).

〔그림 2〕 7비트－3바이트 코드체계



(a) 화면표시



(b) 내부 기억상태

이 코드체계 역시 현재 거의 사용되지 않고 있으며, 16비트 조합형 코드체계로 발전해 가는 중간단계로서 의미를 갖고 있을 뿐이다.

(4) 16비트 조합형 코드체계

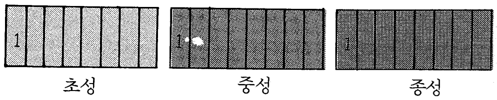
8비트-3바이트 코드체계에서부터는 이제 코드만 보면 한글인지 영문인지 구별이 가능해졌다. 또한, 한글 시작코드와 한글 끝코드를 사용하지 않으므로써 화면에는 보이지도 않으면서 쓸데없이 내부 기억공간만 차지하는 제어코드도 없어졌다. 이쯤이면 한글코드체계도 많은 진보를 한 것이다.

하지만 그래도 문제점은 남아 있다. 완전에 도달하기 위한 마지막 관문이라고 해야 할까? 그것은 내부에 기억된 코드의 길이와 화면이나 프린터로 출력되는 형태에는 차이가 있다는 것이다. 영문은 1바이트이지만 한글 한 자는 내부에 기억될 때에 3바이트를 차지한다. 그러나 화면이나 프린터에 출력될 때에는 한글은 영문의 2배 만큼의 공간을 차지한다. 이 문제를 해결하기 위해서는 한글은 역시 2바이트로 표시할 수 있어야 한다.

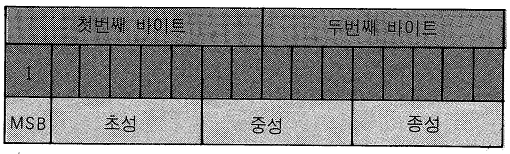
그렇다면 어떻게 3바이트를 2바이트로 줄일 것인가 하는 것이 문제의 관건이 된다. 당시 일본이나 대만에서는 자기들의 문자를 표현하기 위하여 우리가 지금 16비트 완성형이라고 부르는 그네들 나름대로의 코드체계를 가지고 있었다. 한글보다도 훨씬 글자수가 많은 한자도 2바이트로 표시하고 있는데 한글이 불가능할 리는 없는 것이다. 그렇다면 우리의 한글은 어떻게 2바이트로 표시될 수 있을 것인지 **그림 4**를 살펴봄으로써 쉽게 이해할 수 있을 것이다.

지금이야 16비트 조합형하면 이미 귀에 익은지라 별로 대수롭지 않게 여길지도 모르지만, 앞서 언급했던 코드체계들은 16비트 조합형 코드체계가 나오게 된 배경을 설명한 것에 불과하다. 그러면 이 코드체계가 앞서 언급했던 코드체계들에 비하

〔그림 3〕 8비트－3바이트 코드체계



〔그림 4〕 16비트 조합형 코드체계



여 어떠한 장점들을 지니고 있는지 알아보자.

① 한글을 2바이트로 표시하기 때문에 기존의 코드 체계에 비하여 기억장소를 적게 차지한다.
② 내부에 기억되는 코드와 화면이나 프린터에 출력되는 모양이 일반적으로 일치하므로 워드프로세서나 기타 응용 프로그램에서 데이터를 처리하기가 용이하다.
③ 데이터를 검색하거나 분류하는데 있어서 아무런 어려움이 없다.
④ 한글외에도 특수부호나 한자를 위한 코드영역을 쉽게 마련할 수 있다.

16비트 조합형 코드체계도 초성, 중성, 종성을 어떻게 배정하느냐에 따라 여러가지 경우가 있을 수 있다. 가장 널리 알려진 상용 조합형을 비롯해서, 삼성 SPC 조합형, 금성 Mighty 조합형, 그리고 KS 조합형 코드를 표 1에 보였다. 상용 조합형은 우리가 많이 사용하고 있는 NKP와 한국 마이크로소프트의 조합형 한글 MS-DOS(지금은 시판되지 않고 있음)에서 사용하고 있는 코드이다. 삼보컴퓨터의 NKP(또는 THP)와 보석글, 그리고 한글 모듈을 장착한 엡슨 프린터의 보급은 허큘리스(Hercules) 카드를 가진 사용자가 다른 하드웨어의 추가없이 한글을 사용할 수 있게 함으로써 상용 조합형 한글코드를 보급하는데 크게 기여하였다. 그래서 일부에서는 이 코드체계를 삼보 코드라고 부르기도 하는데 이 코드체계는 삼보컴퓨터의 독자적인 것이 아닐뿐만 아니라 큐닉스, 현대전자의 한글카드나 대우 KLDC 한글카드에서도 이를 사용하고 있기에 여기에서는 상용 조합형 코드라고 부르기로 한다.

삼성 SPC 조합형은 SPC-3000 시리즈에서, 금성 Mighty 조합형은 금성의 Mighty 시리즈에서 사용하는 코드를 의미한다. KS 조합형은 여러가지 단점으로 인하여 업계에서 외면당하여 거의 사용되지 않고 있는 코드인데, 작년 1월쯤인가 포항공대에 갔다가 어느 호스트 컴퓨터에 있는 도서관리 프로그램이 이 코드체계를 사용하고 있던 것을 본 적이 있어 굳이 여기에 포함시키기로 하였다.

16비트 조합형 코드체계라 해서 모두가 다 좋은 것은 아니다. 중요한 것은 초성, 중성, 종성을 얼마나 합리적이고 짜임새있게 배치하느냐에 따라 달려 있다. 초성, 중성, 종성을 임의로 배치하다 보면 한글이 제어코드와 겹치게 되어 많은 혼란을 야기시키게 될 뿐만 아니라, 특수부호와 한자까지 사용할 수 있게 하려면 초성 영역의 일부를(가능한 한 앞부분보다는 뒷부분을) 비워두어야만 한다. 따라서 초, 중, 종성의 배치문제는 충분한 검토와 연구가 선행되어야 하며 또한 배치후에도 철저한 시험과정을 거쳐서 가장 효율성 있고도 완전한 체계를 만들어내야 한다. KS 조합형은 초, 중, 종성을 배치하는데 있어서 영문의 제어코드나 OS에서 사용하는 기호문자(*, ?, +, /, \, 〈, 〉 등)들과 겹치지 않도록 해야 하는데 있어서 주의를 게을리하였을 뿐만 아니라 신 코드의 경우에는 한자와 특수부호를 사용할 수 있는 코드영역이 거의 없다는 지적

〔표 1〕 각종 16비트 조합형 한글코드

비트 43210 (Hex)	초 성	중 성	종 성
00000 (00)	FILL		FILL
00001 (01)	ㄱ		ㄱ
00010 (02)	ㄲ	ㅏ	ㄲ
00011 (03)		ㅐ	ㄳ
00100 (04)	ㄴ	ㅑ	ㄴ
00101 (05)		ㅒ	ㄵ
00110 (06)		ㅓ	ㄶ
00111 (07)	ㄷ	ㅔ	ㄷ
01000 (08)	ㄸ		
01001 (09)	ㄹ		ㄹ
01010 (0A)		ㅕ	ㄺ
01011 (0B)		ㅖ	ㄻ
01100 (0C)		ㅗ	ㄼ
01101 (0D)		ㅘ	ㄽ
01110 (0E)		ㅙ	ㄾ
01111 (0F)		ㅚ	ㄿ
10000 (10)			ㅀ
10001 (11)	ㅁ		ㅁ
10010 (12)	ㅂ	ㅛ	ㅂ
10011 (13)	ㅃ	ㅜ	
10100 (14)		ㅝ	ㅄ
10101 (15)	ㅅ	ㅞ	ㅅ
10110 (16)	ㅆ	ㅟ	ㅆ
10111 (17)	ㅇ	ㅠ	ㅇ
11000 (18)	ㅈ		ㅈ
11001 (19)	ㅉ		
11010 (1A)	ㅊ	ㅡ	ㅊ
11011 (1B)	ㅋ	ㅢ	ㅋ
11100 (1C)	ㅌ	ㅣ	ㅌ
11101 (1D)	ㅍ		ㅍ
11110 (1E)	ㅎ		ㅎ
11111 (1F)		FILL	

(a) 금성 Mighty 조합형

을 받고 있다.

(5) 16비트 완성형 코드체계

한글코드체계의 발전단계에서만 보자면 16비트 완성형 코드체계는 조합형 코드체계보다 시기적으로 나중에 나타났다. 하지만 우리가 조합형 한글코드 체계를 사용하고 있을 때에 이미 대만이나 일본에서는 한자를 표현하기 위하여 완성형 코드체계를 사용하고 있었기 때문에 실은 완성형 코드체계가 더 먼저 생겨났다고 보아야 한다.

한자는 한글과는 달리 어떤 하나의 일관된 원리하에 코드를 배정할 수 있는 성질의 것이 아니다. 조합형 코드체계를 사용하더라도 한자만은 완성형을 사용할 수 밖에 없는 이유가 바로 여기에 있다. 완성형 코드에서는 개개의 글자와 코드는 서로 1:1의 대응관계를 가지고 있을뿐 그 이상의 특별한 의미는 없다.

이 코드체계의 특징과 문제점은 무엇인지 알아보기로 하자.

비트 43210 (Hex)	초 성	중 성	종 성
00000 (00)	FILL		FILL
00001 (01)	ㄱ		ㄱ
00010 (02)	ㄲ	FILL	ㄲ
00011 (03)	ㄴ	ㅏ	ㄳ
00100 (04)	ㄷ	ㅐ	ㄴ
00101 (05)	ㄸ	ㅑ	ㄵ
00110 (06)	ㄹ	ㅒ	ㄶ
00111 (07)	ㅁ	ㅓ	ㄷ
01000 (08)	ㅂ		ㄹ
01001 (09)	ㅃ		ㄺ
01010 (0A)	ㅅ	ㅔ	ㄻ
01011 (0B)	ㅆ	ㅕ	ㄼ
01100 (0C)	ㅇ	ㅖ	ㄽ
01101 (0D)	ㅈ	ㅗ	ㄾ
01110 (0E)	ㅉ	ㅘ	ㄿ
01111 (0F)	ㅊ	ㅙ	ㅀ
10000 (10)	ㅋ		ㅁ
10001 (11)	ㅌ		ㅂ
10010 (12)	ㅍ	ㅚ	ㅄ
10011 (13)	ㅎ	ㅛ	ㅅ
10100 (14)		ㅜ	ㅆ
10101 (15)		ㅝ	ㅇ
10110 (16)		ㅞ	ㅈ
10111 (17)		ㅟ	ㅊ
11000 (18)			ㅋ
11001 (19)			ㅌ
11010 (1A)		ㅠ	ㅍ
11011 (1B)		ㅡ	ㅎ
11100 (1C)		ㅢ	
11101 (1D)		ㅣ	
11110 (1E)			
11111 (1F)			

(b) 삼성 SPC-3000 조합형

① 이 코드체계는 16비트 조합형 코드체계와 마찬가지로 내부 기억장소를 차지하는 것과 화면이나 프린터로 표시되는 것이 일치한다.

② 완성형 코드체계는 한글의 특성이나 제자원리와는 전혀 무관한 코드체계로서 한자같은 상형문자나 부호문자를 표현하는데에 접합하며 원래부터 한글과는 어울리지 않았다.

따라서 코드만 보아서는 초성이 무엇인지 또는 모음이나 받침이 무엇인지 알 수가 없다.

요즈음 들어 한글 분야에 있어서도 지능적인 소프트웨어의 필요성을 느낀다. 워드퍼펙과 같은 영문 워드프로세서를 볼 때에 가장 부러움을 느끼게 하는 것이 철자를 검사해 주는 기능이다. 단순히 사전에 없는 단어만을 찾아주는 것이 아니라, 사용자가 어떤 단어를 입력하려고 하다가 철자가 틀렸을 경우에도 스스로 파악하여 예상되는 단어를 나열해 준다. 그 기민함과 영특함에 혀를 내두르지 않은 사람은 아마도 없었을 것이다.

한글 철자검사를 해주는 프로그램을 제작하려면

비트 43210 (Hex)	초 성	중 성	종 성
00000 (00)	미사용	사용제한	사용제한
00001 (01)	FILL	사용제한	FILL
00010 (02)	ㄱ	FILL	ㄱ
00011 (03)	ㄲ	ㅏ	ㄲ
00100 (04)	ㄴ	ㅐ	ㄳ
00101 (05)	ㄷ	ㅑ	ㄴ
00110 (06)	ㄸ	ㅒ	ㄵ
00111 (07)	ㄹ	ㅓ	ㄶ
01000 (08)	ㅁ	사용제한	ㄷ
01001 (09)	ㅂ	사용제한	ㄹ
01010 (0A)	ㅃ	ㅔ	ㄺ
01011 (0B)	ㅅ	ㅕ	ㄻ
01100 (0C)	ㅆ	ㅖ	ㄼ
01101 (0D)	ㅇ	ㅗ	ㄽ
01110 (0E)	ㅈ	ㅘ	ㄾ
01111 (0F)	ㅉ	ㅙ	ㄿ
10000 (10)	ㅊ	사용제한	ㅀ
10001 (11)	ㅋ	사용제한	ㅁ
10010 (12)	ㅌ	ㅚ	미사용
10011 (13)	ㅍ	ㅛ	ㅂ
10100 (14)	ㅎ	ㅜ	ㅄ
10101 (15)	미사용	ㅝ	ㅅ
10110 (16)	미사용	ㅞ	ㅆ
10111 (17)	미사용	ㅟ	ㅇ
11000 (18)	미사용	사용제한	ㅈ
11001 (19)	미사용	사용제한	ㅊ
11010 (1A)	미사용	ㅠ	ㅋ
11011 (1B)	미사용	ㅡ	ㅌ
11100 (1C)	미사용	ㅢ	ㅍ
11101 (1D)	미사용	ㅣ	ㅎ
11110 (1E)	미사용	미사용	미사용
11111 (1F)	사용제한	미사용	사용제한

(c) 상용 조합형

우선 한글의 문법에 대해서 연구해야 한다. 프로그램 스스로가 어느 정도 지능을 갖추어서 체언과 조사가 결합되어 있는 것을 구분할 수 있고, 어간과 어미를 분리해낼 수 있으려면 음절내에서 자소의 구분이 가능해야 한다. 예를 들면, "할지라도"의 경우 "하"+"ㄹ지라도"로 구분할 수 있어야 하는데 완성형 코드에서 이러한 구분이 가능한가?

③ 한글은 모두 자음 19개, 모음 21개, 받침 27개로 조합되어 표시되므로 19×21×28(받침이 없는 경우 포함)=11,172자의 음절을 구성할 수 있다. 그런데 우리의 표준이라고 하는 이 완성형 코드체계에서는 한글의 글자수를 2,350자로 제한하고 있어 이 코드체계에 포함되지 않은 글자들은 우리가 사용할 수 없다는 문제점이 있다.

비트 43210 (Hex)	초 성	중 성	종 성
00000 (00)			FILL
00001 (01)		FILL	ㄱ
00010 (02)		ㅏ	ㄲ
00011 (03)		ㅐ	ㄳ
00100 (04)			
00101 (05)		ㅑ	ㄴ
00110 (06)		ㅒ	ㄵ
00111 (07)		ㅓ	ㄶ
01000 (08)			ㄷ
01001 (09)	FILL	ㅔ	ㄹ
01010 (0A)	ㄱ	ㅕ	ㄺ
01011 (0B)	ㄲ	ㅖ	ㄻ
01100 (0C)	ㄴ		ㄼ
01101 (0D)	ㄷ	ㅗ	ㄽ
01110 (0E)	ㄸ	ㅘ	ㄾ
01111 (0F)	ㄹ	ㅙ	ㄿ
10000 (10)	ㅁ		ㅀ
10001 (11)	ㅂ	ㅚ	ㅁ
10010 (12)	ㅃ	ㅛ	ㅂ
10011 (13)	ㅅ	ㅜ	ㅄ
10100 (14)	ㅆ		ㅅ
10101 (15)	ㅇ	ㅝ	ㅆ
10110 (16)	ㅈ	ㅞ	ㅇ
10111 (17)	ㅉ	ㅟ	ㅈ
11000 (18)	ㅊ		ㅊ
11001 (19)	ㅋ	ㅠ	ㅋ
11010 (1A)	ㅌ	ㅡ	ㅌ
11011 (1B)	ㅍ	ㅢ	ㅍ
11100 (1C)	ㅎ		ㅎ
11101 (1D)		ㅣ	
11110 (1E)			
11111 (1F)			

(d) KS 조합형(구형)

이 문제를 보완하기 위한 편법으로 코드영역중 일부를 사용자 정의 문자를 위해 비워두긴 했지만, 이것은 근본원칙을 무시함으로써 비롯된 문제를 우회적으로 무마하려 한 것이기에 근원적인 해결책이 될 수가 없음을 보여주고 있다.

다음은 사용자 정의 문자라는 편법으로 인하여 생겨나는 문제점들이다.

a. 사용자 정의영역에 관한 한, 이것은 사용자 개개인이 자의대로 2,350자에 없는 글자중에서 자신이 필요로 하는 글자를 정의해서 사용할 수 있게 한 것이므로 이 부분에 대한 통일을 기대할 수가 없다는 것이다. 즉, 통일을 위한 표준안은 통일을 하지 않기 위한 표준안이라는 지적을 받게 되는 것이다.

b. 사용자 정의영역은 코드의 순서로 볼 때 한글코

비트 43210 (Hex)	초 성	중 성	종 성
00000 (00)			FILL
00001 (01)	ㄱ		ㄱ
00010 (02)	ㄲ	ㅏ	ㄲ
00011 (03)		ㅐ	ㄳ
00100 (04)	ㄴ	ㅑ	ㄴ
00101 (05)		ㅒ	ㄵ
00110 (06)		ㅓ	ㄶ
00111 (07)	ㄷ	ㅔ	ㄷ
01000 (08)	ㄸ		
01001 (09)	ㄹ		ㄹ
01010 (0A)		ㅕ	ㄺ
01011 (0B)		ㅖ	ㄻ
01100 (0C)		ㅗ	ㄼ
01101 (0D)		ㅘ	ㄽ
01110 (0E)		ㅙ	ㄾ
01111 (0F)		ㅚ	ㄿ
10000 (10)			ㅀ
10001 (11)	ㅁ		ㅁ
10010 (12)	ㅂ	ㅛ	ㅂ
10011 (13)	ㅃ	ㅜ	
10100 (14)		ㅝ	ㅄ
10101 (15)	ㅅ	ㅞ	ㅅ
10110 (16)	ㅆ	ㅟ	ㅆ
10111 (17)	ㅇ	ㅠ	ㅇ
11000 (18)	ㅈ		ㅈ
11001 (19)	ㅉ		
11010 (1A)	ㅊ	ㅡ	ㅊ
11011 (1B)	ㅋ	ㅢ	ㅋ
11100 (1C)	ㅌ	ㅣ	ㅌ
11101 (1D)	ㅍ		ㅍ
11110 (1E)	ㅎ		ㅎ
11111 (1F)			

(e) KS 조합형(신형)

드 영역의 뒷부분과 한자코드 영역의 뒷부분 두 군데에 위치하고 있다. 이는 한글코드 영역의 뒷부분에 있는 코드영역에는 2,350자외의 한글을 정의하여 사용하고, 한자코드 영역 뒷부분의 코드 영역은 4,888자외의 한자를 위하여 마련한 듯하다.

만약 "넋"자를 입력하려고 하는데 "넋"자는 2,350자에 속하지 않으므로, 이것을 입력하기 위해서는 사용자 정의문자로 등록시켜야 한다.(지금 이순간 나의 컴퓨터가 완성형 코드를 사용하는 행정전산망용이 아니라는 것이 왜 이리도 다행스럽고 즐거운지 모르겠다.) 그런데 이 글자는 사전순서상으로 볼 때에 분명 "힝"자보다는 앞쪽에 있음에도 불구하고 코드상으로는 "힝"자보다 뒷쪽에 위치하게 된다. 이런 경우 우리는 자료를 분류하는데 있어서 얼마나 애를 먹어야 되는걸까?

c. 사용자 정의문자를 표시하기 위해서는 한글카드에 별도의 램(RAM)을 설치하여 사용자 정의문자의 자형을 다운로드하여야 하는 점도 있다.

④ 한글카드를 설계하는데 있어서도 완성형 코드를 사용하는 한글카드는 조합형 코드를 사용하는 한글카드보다 제작비가 더 들게 된다. 조합형 코드는 아무런 변환과정 없이도 곧바로 조합형 자형을 사용하여 출력할 수 있지만, 완성형 코드의 경우에는 완성형 자형을 사용하든지 아니면 조합형 코드로 변환한 다음에 조합형 자형을 사용하든지 해야 한다. 조합형 자형과 완성형 자형이란 무엇인지 그 특징에 관하여는 뒷부분의 한글 출력에 관한 부분에서 언급하기로 한다.

(6) 코드문제에 붙여서

이외에도 흔히들 세운상가 한글카드라고 부르는 7비트 2바이트 완성형 코드체계가 있는데 이는 16비트 완성형 코드보다도 훨씬 엉터리 코드체계이며, 정상적인 한글문화의 발전을 위하여 가급적 빨리 이 땅에서 추방되어야 할 것이라고 생각하기에 여기에서는 일언반구조차도 거론하지 않기로 한다.

표준코드는 절대로 자주 개편되어서는 안된다. 표준이 자주 바뀐다는 것은 관련분야에 있어서 뿐만 아니라 간접적으로나마 연관이 있는 다른 분야에까지도 파급적으로 영향을 끼치기 때문에 상당한 혼란을 가져온다. 더구나 우리 모두가 사용하고 있는 문자에 관한 표준이라면 이것은 컴퓨터에 관련된 문제가 아니라 우리의 일상 생활과 직결된 문제이므로 그 문제는 더욱 심각할 것이다.

표준코드의 개정은 맞춤법의 개정과는 다른 면이 있다. 맞춤법이 바뀌었다고 해서 이미 출판된 모든 서적을 다시 인쇄해야 하는 것은 아니다. 하지만 표준코드가 개정되면 기존의 데이터는 새로운 코드로 변환을 시켜주어야 하며, 하드웨어 역시 새로 개발하여야 하므로 그로 인한 비용은 결코 무시할 수 없다. 표준이 자주 바뀌지 않으려면 만들 때 잘 만들어야 한다. 엉터리로 만들어 놓은 표준을 누가 쉽사리 따르려고 노력할 것이며 또한 그 생명은 얼마나 지속될 수 있을 것인가?

우리의 언어생활은 항상 사용하는 단어들의 반복으로만 이루어지는 것이 아니다. 우리의 삶은 항상 새로운 것을 추구해 나가고 있으며 오늘은 결코 어제의 반복이 아니다. 언어생활 역시 마찬가지이며 우리는 항상 새로운 말들을 만들어내고 있으며, 더 나은 자기표현을 위하여 노력한다. 그러므로 언어란 정적인 것이 아니라 활동력을 가지고 있는 것이라면 나의 지나친 억측일까?

'Word power made easy'라는 영어 단어 학습책의 서두에 보면 우리의 사고능력은 얼마만큼의 어휘를 알고 있느냐에 좌우된다고 한다. 10,000자가 넘는 글자를 2,350자로 제한시키고 있는 것은 우리의 폭넓은 언어생활을 해치고 있을 뿐만 아니라 더 나아가서는 우리의 사고능력에까지 제약을 가하고 있는 것이다.

한글과 완성형 코드는 이루어질 수 없는 관계이다. 필자가 알기로는 국민학교 교육도 못받은 사람은 높은 자리에 앉을 수 없다고 알고 있는데 요즈음은 그게 아닌지도 모른다. 마치 한글이 소리나는 대로 다 적지 못하는 것이 현실인 것처럼… 행정관계 당국이나 정책입안자들은 좀 각성을 하여야 한다. 행정은 누구를 위해서 하는 것일까? 잘못된 행정으로 인하여 피해를 입는 것은 결국 누구인가? "국민을 위하여"란 말을 하기 전에 "국민을 괴롭히지 않기 위한"을 먼저 실천해 주었으면 한다.

한글 입력자판

한글코드 문제와 더불어 또 한가지 빠뜨리고 갈 수 없는 것이 바로 한글 자판배열에 관한 문제이다. 우선 한글자판이 어떻게 변천해 왔는지 간략하게 알아보기로 하자.

(1) 한글 자판배열의 변천

한글 자판배열의 역사를 더듬어 보자면 한글 타자기가 발명된 때로 거슬러 올라가지 않으면 안된다. 즉, 글자판의 시초는 수동식 타자기에서부터 시작되고 이후에 전동식 타자기나 컴퓨터로 발전해 온 것이다. 맨 처음 수동식 타자기를 발명한 사람은 안과의사 공병우 박사였다. 공병우 박사는 한글의 구조에 따라 자판배열을 3벌식(초성한벌, 중성한벌, 종성한벌)으로 하였는데, 그 이후에 글자의

5벌식 구조를 채택하여 글자의 모양을 좀 예쁘게 만든 김동훈식 타자기가 나왔다.

나중에 이를 통일하기 위하여 박정희 정권때에 과학기술처가 두 글자판의 장점을 따서 만든답시고 한 것이 4벌식이었다. 이는 초성 한벌, 받침의 유무에 따라 중성 두벌, 종성 한벌의 구조로 되어 있는데, "공병우식과 김동훈식의 단점만 모은 졸작"(조선일보 1977.9.28)이라는 평을 받았으며, 3벌식 주장자들은 그 이후로 계속 자판통일을 위한 논쟁을 벌여왔다. 그러다가 5공화국 시절에 새로운 표준자판을 정한 것이 2벌식인데 이는 4벌식보다도 더 엉터리였다. 그래서 송현(한글기계화추진회 회장) 어른같은 분은 이 2벌식 글자판의 제정은 5공화국 비리중 가장 큰 비리에 속한다고 말하고 있다.

(2) 2벌식 자판배열의 문제점

글자판의 논쟁은 어제 오늘의 이야기가 아니며, 엉터리, 글자판이 표준으로 존속하는 한 그 논란은 끊임없이 계속될 것이다. 그렇다면 두벌식 글자판은 무엇이 문제이며, 이것이 컴퓨터에 관련하여 어떠한 문제들이 있는지 알아보도록 하자.

2벌식 자판의 단점은 3벌식 자판의 장점이 되므로 따로 언급하지 않기로 하며 과연 어느 자판배열이 우리의 한글을 입력하는데 더 적합한지는 여러분 스스로가 판단할 일이다.

① 우선 속도문제를 들 수 있다. 수동식 타자기에 있어서는 받침은 모두 시프트 키를 누른 상태에서 입력해야 하므로 손가락이 힘이 들뿐만 아니라 그로 인하여 타자속도도 느려지는 것이다. 또한, 속도문제에 큰 영향을 끼치는 요인으로서 2벌식 자체가 지니는 구조상의 문제이다. 한글은 초성, 중성, 종성으로 되어 있어서 같은 자음일지라도 초성자음과 받침은 전혀 다른 것이며 그 둘을 같은 것으로 취급하여야 할 아무런 이유도 없다. 그런데 2벌식 글자판에서는 그 둘을 하나의 키에 배정함으로써 마치 똑같은 자음인 것처럼 취급하고 있다.

"국가"라는 말은 2벌식으로 입력한다고 가정해 보자. 우리는 "ㄱㅜㄱㄱㅏ"라고 치게 되는데, 처음 타자를 배우는 사람의 경우는 이것을 머리속에서 하나씩 뇌이면서 타자를 하게 된다. 즉, 머리속에서 자신이 알게 모르게 풀어쓰기를 하는 셈이다. 이 현상은 원래 우리가 같지 않다고 인식하고 있던 초성 "ㄱ"과 받침 "ㄱ"을 의도적으로 똑같은 것인양 동일시하려는데서 빚어진 쓸데없는 노력이다.

이처럼 2벌식 자판에서는 타자를 하면서 의식적이든 무의식적이든 초성 "ㄱ"과 받침 "ㄱ"을 같은 것으로 해석해내야 하는 과정이 속도를 떨어뜨리는 주요한 원인으로 작용하고 있는 것이다. 마치 우리가 독서를 빨리하기 위해서는 머릿속으로 따라읽기 하는 습관을 버려야 하는 이치와도 같은 것이다.

② 컴퓨터에 있어서 한글 입력 오토메타에서 자음이 입력되었을 때에 초성과 종성을 구분하여 처리해 주어야 하는 어려움이 따른다. 또한, 이것은 우리가 의도하지 않는 글자를 중간에 보여주다가 갑자기 변화시키는 기이한 현상을 초래하여야만 한다. 가령 "우리나라"를 입력할 때에 이것이 화면에 어떻게 표시되는지 한번 보자.

ㅇ→우→울→우리→우린→우리나→우리날
→우리나라

위의 같은 현상을 이제 한글을 배우는 아이들에게 어떻게 설명할 것인가? 하지만 3벌식에서는 우리가 연필로 글씨를 쓰듯이 나타낼 수 있다.

ㅇ→우→우ㄹ→우리→우리ㄴ→우리나
→우리나ㄹ→우리나라

어느 것이 올바른 것인지는 독자 여러분이 더 잘 알 것이다.

③ 한글의 받침만을 초성자음과 구별하여 입력할 수가 없다. 위의 ①항에서 "ㄱ"과 "ㄱ"을 보였었는데 뒤의 것은 받침이기 때문에 앞의 것과는 모양이 달라서 아랫쪽으로 치우져 있다. 지금 현재 필자가 2벌식 자판배열을 사용하고 있다면 이것 둘을 어떻게 구별하여 입력할 수 있을 것인가? 특히 표준 완성형 코드에서는 "ㄱ"은 하나만 둠으로써 아예 이 둘의 구별을 인정하지 않고 있다.

④ 2벌식 글자판은 3벌식에 비하여 오타가 많이 발생한다. 이는 인간공학적인 면에서 왼손에 비하여 오른손을 더 많이 사용하도록 하여야 한다든지, 자주 사용하는 자모를 누르기 쉬운 위치에 배열하여야 한다든지, 자모간의 결합관계 및 비율을 따져서 가급적 손가락을 덜 움직이도록 하여야 하는 인간공학적인 면을 무시한 결과이다.

⑤ 마지막으로 2벌식 글자판은 통일을 위한 글자판이 아니라 통일을 하지 않기 위한 글자판이라는 점

이다.

이 경우에 있어서는 필자의 경험담이 대표적인 예가 될 수 있겠다. 필자는 타자기보다는 컴퓨터를 먼저 만지게 된 세대에 속하는데, 막 군입대를 하고 논산훈련소에서 있었던 에피소드 한토막이다.

하루는 중대장이 타자를 칠 수 있는 사람을 몇 명 뽑아오라고 하였다. 당시 필자는 컴퓨터용 2벌식 자판에는 자신이 있다고 생각하고 있던 터였으므로 자신있게 지원자에 가담하였다. 모두 뽑혀간 사람이 네명이었었는데 다들 중대장이 내놓은 타자기는 쳐보지 않은 것이라며 물러서는 것이었다. 가만 보니 그 타자기는 바로 2벌식이었으며 다들 4벌식 타자기만 사용해온 까닭이다. 필자는 그것이 컴퓨터 자판배열과 같아 보여서 응당할 수 있으리라 생각하고 열심히 해 보았는데 도무지 손가락이 잘못된 것인지 글자가 이상하게 나타나고 원하는대로 되지 않는 것이었다. 가만보니 이것은 받침을 칠때마다 시프트 키를 누른 상태에서 쳐야 하는데, 컴퓨터에서는 자음은 쌍자음이 아니면 시프트 키를 사용하지 않는다. 그 엄청난(?) 차이로 인하여 2벌식 수동타자기는 컴퓨터와 똑같은 2벌식 배열이면서도 필자에게는 아예 사용할 수 없는 다른 자판이 되어 버린 것이다. 분명 같은 2벌식 자판인데도 컴퓨터를 사용하는 사람은 타자기를 사용할 수 없고, 타자기를 사용하는 사람은 컴퓨터를 사용할 수 없단 말인가? 영문 타자를 할 수 있는 사람이면 누구나 컴퓨터로도 영문 타자를 할 수 있는 사실과 비교해 보면, 우린 이것이 글자판의 통일인지 아니면 분열인지 알 수 있을 것이다.

한글입력 오토메타

(1) 한글입력 오토메타란?

오토메타 이론은 원래 컴파일러가 소스코드를 해석하는 과정에서 발전된 것이다. 한글 입력 오토메타란 키보드에서 입력되는 문자열(음소)을 분석하고 이를 음절별로 구성하여 해당 음절의 코드를 발생시키기 위한 알고리즘 및 프로그램을 뜻한다. 영문은 하나의 키입력이 하나의 문자코드를 발생시키지만 한글의 경우에는 초성, 중성, 종성이 결합되어 한 글자를 이루기 때문에 하나의 글자를 입력하기 위해서는 여러번의 키입력이 필요하다. 한글 입력시에 모아쓰기를 해주는 오토메타는 현재 한글 2000시리즈로 유명한 '한 컴퓨터 연구소'(개발 당시에는 아직 연구소 설립이 되지 않았음)가 개발한 애플 컴퓨터의 한글 워드프로세서에서 처음 선보인 것으로 알려지고 있다.

(2) 한글 입력 오토메타 작성에 있어서 선결되어야 할 것들

한글 입력 오토메타는 어느 코드체계를 사용하느냐에 따라서, 그리고 자판배열이 어떻게 되느냐에 따라서 크게 달라진다. 아직까지 우리나라에는 통일된 한글 입력 오토메타가 없다. 행정전산망용 입력 오토메타의 규격이 정해져 있긴 하지만, 그것은 완성형 한글코드를 위한 오토메타이므로 행정전산망용 컴퓨터 이외의 다른 16비트 개인용 컴퓨터에 두루 적용하기는 어렵다.

한글 입력 오토메타를 구성하기 위해서는 다음과 같은 문제들이 먼저 해결되어야 한다.

① 한글코드는 어느 것을 채택하여야 할 것인가 하는 문제이다. 이것은 오토메타에서 키입력을 해석하여 한글코드를 발생시킬 때에 어느 코드체계의 한글코드를 생성해낼 것인가에 관한 문제이다.

② 한글 자판은 어느 체계를 따를 것인가? 그리고 한글 / 영문 변환키는 어느 것으로 할 것인가? 한 / 영 변환 키는 자주 사용하는 것이므로 누르기 쉬운 위치에 있어야 하며, 가급적이면 두 개의 키를 조합하여 사용하는 것보다는 하나의 키로서 사용할 수 있도록 하는 것이 바람직하다.

그러나 하나의 키를 사용하는 경우에 있어서 행정전산망용 PC에서와 같이 한 / 영 변환키가 따로 존재하지 않는 일반 컴퓨터의 자판에서는 어느 특정 키를 따로 한 / 영 변환키로 설정하면 그 키의 본래의 기능을 사용할 수 없게 되는 문제점이 발생한다. 그리고 한글처리 프로그램 중에는 한 / 영 변환키가 토글방식(하나의 키로서 전환하는 방식)이 아니라 한글모드를 위한 키와 영문모드를 위한 키가 따로 존재하는 것도 있는데, 이 방식은 그 프로그램에서 현재의 입력상태가 한글 입력상태인지 영문 입력상태인지 구분할 수 있는 방법을 제공하지 않은데서 연유한다.

③ 입력 가능한 자모의 범위를 선정하는 문제이다. 우리 주위에서 사용되고 있는 한글처리 프로그램에서는 입력가능한 자모의 범위를 어떻게 제한하고 있는지 알아보도록 하자. 아래에서 모아쓴 한글이란 초성+중성, 또는 초성+중성+종성으로 결합된 글자를 말하며, 단자음은 "ㄱ", "ㄴ"같은 하나의 자음을, 쌍자음은 "ㄲ", "ㄸ" 같은 자음을, 복자음은 "ㅄ", "ㄺ" 같이 받침으로만 사용되는 자음을 말하며, 단모음은 "ㅏ", "ㅑ", "ㅓ", "ㅕ" 등의 기본 10개 모음에 "ㅐ", "ㅔ", "ㅒ", "ㅖ" 4개를 포함한 14개의 모음을, 중모음은 "ㅢ", "ㅘ" 같이 두개의 모음이 결합된 것을 말한다.

a. THP, NKP, 조합형 한글 MS-DOS : 모아쓴 한글, 단자음, 쌍자음, 단모음

b. 완성형 한글 MS-DOS(행정전산망용) : 모아쓴 한글, 단자음, 쌍자음, 복자음, 단모음, 중모음
c. 이찬진씨의 한글 워드프로세서 : 모아쓴 한글, 단자음, 쌍자음, 복자음(3벌식 자판에서 지원), 단모음, 중모음, 중성+종성의 한글(초성없이 모음과 받침으로만 구성된 글자), 모음없이 초성+종성으로만 이루어진 글자(3벌식 자판에서 지원)

④ 'BS'(Back Space)키의 처리방법에 관한 문제이다. 16비트 코드체계에서 한글은 화면이나 기억장소에서 영문의 2배를 차지한다. 'BS'키를 눌렀을 때 한글은 한번에 지울 것인지 아니면 'BS' 두번을 사용해서 지울 것인지 하는 것에 관한 문제이다. 또한, 한글 입력중에 'BS'가 눌린 경우 입력중인 한글(아직 코드가 완성되지 않은 상태의 한글)을 모두 지울 것인지 아니면 마지막 입력된 자소 하나만 지울 것인지 하는 것도 여기에서 함께 고려해야 할 대상이다. 이 문제 역시 우리 주위에서 사용되고 있는 한글처리 프로그램에서 처리하고 있는 방식을 알아봄으로써 가장 보편타당성이 있는 방식을 생각해 보도록 하자.

a. THP, NKP, CHP 구 버전 : 프로그램을 실행시킬 때에 뒤에 옵션 'g'를 붙일 수 있는데, 'g' 옵션이 주어진 경우에는 한글을 지우기 위해서는 'BS'키가 두번 필요하며, 그렇지 않은 경우에는 'BS' 한번만으로 한글을 지울 수 있다. 한글이 입력중일 때에 왼쪽 화살표를 누르면 가장 나중에 입력된 자소 하나만 지울 수 있다.
b. CHP 새 버전 : 뒤에 옵션 'g'를 붙이는 것은 위와 같다. 한글 입력중에 마지막으로 입력된 자소를 지우는 키가 'BS'로 바뀌었다.
c. 조합형 한글 MS-DOS, 완성형 한글 MS-DOS (행정전산망용) : 한글을 지우기 위해서는 'BS'가 두번 필요하며 마지막으로 입력된 자소를 지우는 방법은 없다.
d. 이찬진씨의 한글 워드프로세서 : 한글은 항상 'BS' 한번만으로 지워지며, 한글이 입력중일 경우 'BS'는 마지막으로 입력된 자소를 지운다.

위에서 보다시피, 한글은 영문과는 그 크기가 다르지만 'BS' 한번으로 지울 수 있어야 하고, 또한 입력중인 한글의 경우에도 마지막 입력된 자소를 지울 수 있는 것이 사용자의 불필요한 키입력을 방지해준다는 측면에서 더욱 바람직할 것이다.

그러나, NKP나 한글 MS-DOS와 같이 BIOS나 OS수준에서 한글을 처리하고 있는 경우에는 보석글과 같은 응용 프로그램과의 충돌을 방지하기 위하여 'BS'처리에 있어서 옵션을 두어야 할 것이다. 왜냐하면 보석글과 같은 응용 프로그램에서는 'BS'키가 입력되었을 때에 영문과 한글에 대해서 어떻게 지워야 하는지 스스로 알아서 처리하고 있기 때문이다. 만약 NKP나 CHP에서도 'BS'에 대한 처리를 한다면 결국 두개의 글자가 지워지는 현상이 나타나기 때문이다.

⑤ 종성으로 쓰이는 쌍자음("ㄲ", "ㅆ")을 입력하는데 있어서 단자음 키를 두번 누르는 것을 허용할 것인지 하는 문제이다. 현행 2벌식 자판에서는 쌍자음에는 각 키를 배정했으나 복자음은 두 개의 키를 조합하여 입력하도록 되어 있다. 그러므로 쌍자음에 있어서는 언제든지 하나의 키로서 입력가능하나, 그것이 받침으로 쓰이는 경우에 있어서 복자음의 입력방식과 통일을 기한다는 측면에서 단자음을 두번 누르는 것을 허용할 것인지의 여부를 논하는 문제이다. THP, NKP, CHP 구버전에서는 이를 가능하도록 하고 있는데 그 외의 프로그램들에서는 이러한 입력방식을 인정하지 않고 있는 추세이다.

⑥ Caps Lock 상태(영문 대문자 입력상태)일 때에 한글입력의 가능여부에 관한 문제이다. 한글처리 프로그램 중에는 Caps Lock이 오프된 상태(영문 소문자 입력상태)에서만 한글입력이 가능한 것들이 있다. 한글 입력상태와 영문 입력상태는 전혀 별개의 세계로 구분되며 한글에는 대·소문자 같은 구별이 없으므로, 영문의 대·소문자 입력상태를 전환시켜주는 Caps Lock이 한글입력에 영향을 미친다는 것은 일반적인 상식의 범위에서 벗어난다. 따라서 한글 입력 오토메타에서는 Caps Lock 상태와는 무관하게 한글을 입력할 수 있도록 하는 것이 바람직할 것이다.

⑦ 한글 미완성 글자의 코드에 대해서도 처리하고 있는 지에 관한 문제인데, BIOS나 OS 수준에서 한글처리를 하는 프로그램에 있어서는 아주 중요한 문제라고 생각된다. 우리가 흔히 사용하고 있는 보석글과 팔란티어 워드프로세서를 예로 들어 설명하기로 한다. 삽입모드에서 글자를 입력하는 경우 다음과 같은 상태에 있다고 가정해 보자.

삽입상태에서 한글을 입하는 경우

이때 커서는 "하"에 위치하고 있으며 여기에서 "력"을 입력하면 다음과 같은 상태로 변화된다.

a. 보석글의 경우(THP, NKP, CHP 사용시)

입하는→입근 는→입려는→입력는

위는 받침 "ㄱ"까지 입력하고 난 직후인데 이때 오른쪽 화살표 키를 누르면 "하"가 튀어나오면서 "입력하는"으로 된다.

b. 팔란티어 워드프로세서의 경우(한글 MS-DOS 사용시)

입하는→입근하는→입려하는→입력하는

어느 것이 더 바람직한 것인지는 독자 스스로가 판단할 수 있으리라 믿는다. 왜 이렇게 서로 다른 현상이 일어나는가? 우선 THP, NKP, CHP에서는 한글 미완성코드를 알아볼 수 있는 방법을 제공하지 않고 있다. 그리고 이 프로그램의 한글 입출력 기능을 이용하고 있는 보석글에서는 당연히 아직 완성되지 않은 한글을 처리할 수가 없는 것이다. 반면 한글 MS-DOS에서는 한글 미완성코드도 되돌려 주는 함수(Function)를 제공하고 있으며, 이 기능을 이용하여 팔란티어 워드프로세서는 아주 멋지게 일을 처리하고 있는 것이다.

⑧ 그외에도 현재 입력상태가 한글인지 영문인지 구분가능하게 해주는 것도 한글 입력 오토메타와 관련된 문제에 속하므로 고려해야 할 대상에 속하는데, 현재 사용되고 있는 프로그램들은 다음과 같은 방법을 통해서 입력상태를 확인할 수 있게 하여준다.

a. 화면 아래쪽에 현재 입력상태를 표시해 주는 방법 : K-System, Eten 한글카드, 이찬진씨의 한글 워드프로세서

b. 커서의 모양을 다르게 하는 방법 : CHP 새 버전, 한글 MS-DOS 등 다수

상기와 같은 사항들은 우리가 흔히 접할 수 있는 문제이지만 이외에도 실제 오토메타를 구성하다 보면 예기치 못한 사사로운 문제점들에 가로막힐 경우가 많다. 그럴 경우에는 항상 보편타당성을 유지하면서 통일을 기하는 방향으로 접근해야 한다.

조합형 한글코드를 사용하는 한글처리 프로그램에 있어서도 그 상이한 방식으로 인하여 사용자는 상당한 혼란을 겪고 있다. 실지로 최근 한국경제신문에서 운영하고 있는 BBS(아직 완성된 단계가 아니라서 일반에게 공개되지 않았음)의 게시판에 실린 어느 편지에 의하면, 공용디렉토리에 들어있는 한글처리 프로그램들마다 제각기 한/영 변환키가 달라서 보물찾기 하는 식으로 모든 키를 다 뒤지다시피 하여 간신히 알아냈다고 하소연하고 있다.

한글 입력 오토메타 역시 표준을 제정하여 하나로 통일하여야 한다는 것은 우리 모두가 그 피해자로서 공감하고 있는 일이다. 하지만 오토메타의 표준화는 앞서 언급한 코드와 자판의 통일과는 달리 모든 사람들이 수용할 수 있는 합리적이고 보편타당성을 가졌으면 한다. 그 바램으로 욕심같지만 본고에서는 표준을 지향한 현실성있는 한글 입력 오토메타의 체계를 한번 구성해 볼까 한다.

(3) 한글 입력 오토메타의 구성 및 도해

지금부터 구성해 보고자 하는 한글 입력 오토메타는 자판배열에 따라 2벌식 표준자판용과 공병우 3벌식 자판용 두개로 구분하여 다루기로 한다.

• 표준 2벌식 자판용 한글 입력 오토메타

본고에서 구성하고자 하는 표준 2벌식 자판배열(그림 5 참조)을 채택한 한글 입력 오토메타는 다음과 같은 규격을 갖게 되는데, 이 규격은 필자의 수준에서 가장 타당성이 있다고 생각되는 것들을 선택한 것이므로 더 나은 방법이 있다면 원하는대로 조정하면 될 것이다.

① 한글코드는 상용 조합형을 사용하기로 한다. 코드값을 돌려주기 전에 다른 조합형 코드나 완성형 코드로 변환함으로써 다른 코드도 쉽게 지원할 수 있게 된다. 단, 조합형이 아닌 완성형 코드를 지원하고자 하는 경우에는 각 상태가 종료될 때마다 그 때까지 구성된 한글이 완성형 코드에 속하는지 검사하여, 만약 완성형 코드에 없는 글자이면 나중

〔그림 5〕 표준 2벌식 자판배열



에 입력된 자소는 다음 글자를 구성하는 것으로 처리해 주어야 한다. 이러한 배려를 하지 않은 경우에는 K-System의 경우처럼 완성형 코드에 없는 글자가 만들어지는 순간 화면에서 입력되고 있던 글자가 갑자기 사라지는 현상이 나타나게 된다.

② 한글 입력자판은 현행 컴퓨터용 표준 2벌식을 사용하기로 한다. 한/영 변환키는 Alt키와 스페이스(Space)키를 동시에 누르는 것으로 한다. 두 개의 키를 조합했기 때문에 하나의 키보다는 다소 불편하겠지만, 어느 자판에서나 비슷한 위치에 있고, 비교적 누르기 편리한 위치에 있을 뿐만 아니라 원래 키의 기능을 그대로 사용할 수 있다는 장점이 있으므로 한/영 변환키로서는 무난하리라고 생각한다.

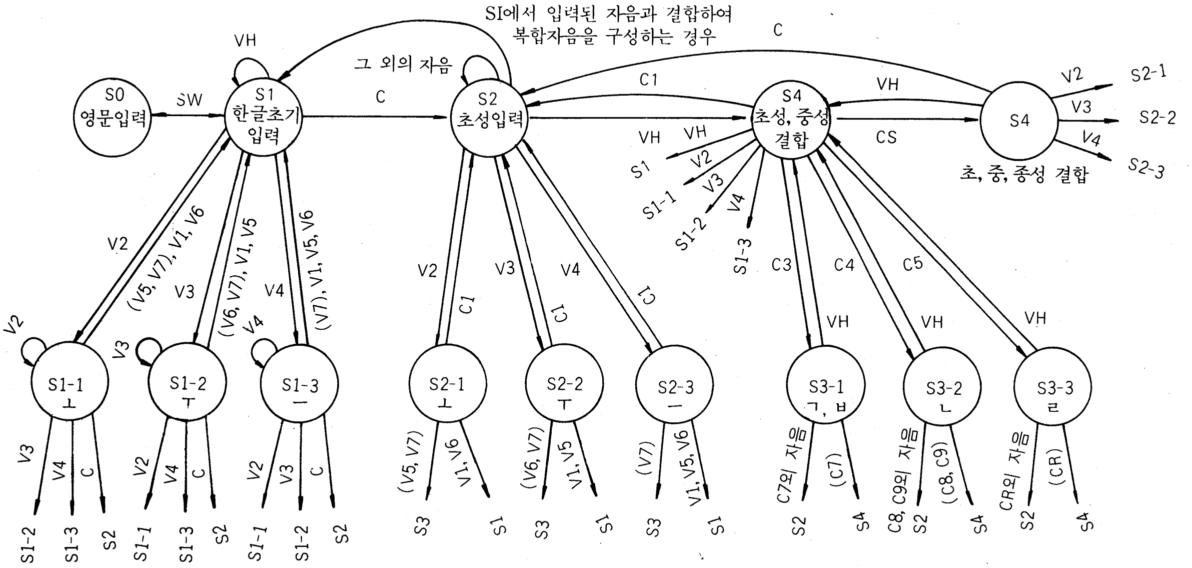
③ 입력가능한 자모의 범위는 모아쓴 한글, 단자음, 쌍자음, 복자음, 단모음, 중모음으로 정한다. 초성없이 모음과 받침만으로 구성된 글자나 모음없이 초성과 종성으로만 이루어진 글자는 우리 문자생활에서 전혀 사용되지 않으며 한글의 제자원리에 비추어 보아도 적합치 않으므로 여기에서는 제외하기로 한다. 이찬진씨의 한글 워드프로세서에서 이를 지원하고 있는 것은 모음을 처리하는 과정을 간결하게 하기 위한 것으로 풀이된다.

이와같은 사실은 모음만 반복해서 입력할 경우의 처리방식이 일관되지 않음을 볼 때에 쉽게 짐작할 수 있다. 본고에서는 아무리 오토메타가 복잡해지더라도 중요 원칙은 존중하며 사소한 사항은 필요에 따라 취사선택하는 입장을 취함에 따라, 입력오토메타의 각 단계별 흐름을 나타낸 그림 6의 상태전이도와 같이 S1-1, S1-2, S1-3의 세개의 상태를 더 두게 되었다.

그리고 복합자음의 입력에 관한 부분은 프로그래밍 과정에서는 서너 줄을 더 추가하는 정도로 처리할 수 있으나, 오토메타의 상태전이도에서는 세 개의 상태를 더 추가하게 하므로 모든 조건을 다 따져서 처리하지 않고 그냥 간략화하여 표시했다.

④ 'BS'키에 대한 처리는 한글 입력도중에는 입력된 순서의 역으로 하나의 음소씩 지울 수 있도록

〔그림 6〕 표준 2벌식 자판배열에 따른 한글 입력오토메타 상태전이도 (p171 참조)



• C= {모든 자음}
CS= {C2, C6, C7, C8, C9}
VH= {V1, V5, V6, V7}
CR= {C3, C6, C7, C9}
• α의 입력은 항상 S1상태로 간다.
• 괄호안의 자음, 모음은 복합자음, 복합모음을 구성하는 것을 나타낸다.

S2-1, S2-2, S2-3 공통처리	
C3 →	S3-1
C4 →	S3-2
C5 →	S3-3
CS →	S4
V2 →	S1-1
V3 →	S1-2
V4 →	S1-3

S3-1, S3-2, S3-3 공통처리	
V2 →	S2-1
V3 →	S2-2
V4 →	S2-3

하고, 입력이 완성된 상태에서는 'BS' 코드를 돌려주도록 한다. 이에 관한 처리 역시 프로그램에서는 스택이라는 자료구조를 통해서 손쉽게 구현할 수 있지만 오토메타 상태전이도에는 막대한 영향을 끼치게 되므로 생략하였다.

⑤ 종성으로 쓰이는 쌍자음은 해당 쌍자음 키를 누를 경우에만 쌍자음으로 처리하고, 단자음의 결합은 인정하지 않는다.

⑥ 한글 입력은 Caps Lock 상태와는 무관하게 이루어진다.

⑦ 한글 미완성 글자에 대한 코드도 돌려줄 수 있도록 한다.

• 프로그램 설명

리스트 1의 프로그램은 가능한 행정전산망용 BIOS 규격과 일치시키고자 하였다. 그러나 행정전산망용 BIOS에서는 확장 키보드 BIOS(Extended Keyboard BIOS)를 지원하지 않고 있으므로 이 모듈을 사용하시려는 분이나 또는 한글처리에 관심을 가진 분을 위하여 확장 키보드 BIOS에 관하여 간략하게 설명드리겠다. 원래의 키보드 BIOS는 표 2에 보인 키보드 BIOS중에서 0, 1, 2 세개의 펑션(Function)만 있었으나 이후에 AT 컴퓨터와 101개의 키를 가진 자판이 나옴에 따라서 기존의 세개의 펑션으로는 부족함을 느끼게 되었다.

특히 101 자판에서는 Ctrl, Alt, Home, End 등의 키가 두개로 늘어나게 됨에 따라 이 두개의 키 중에서 어느 키가 눌렸는지 구분해 줄 필요가 생겼다. 그에 따라 기존의 BIOS와의 호환성을 유지하기 위하여 기존의 세 개의 펑션외에 새로운 펑션들이 추가된 것이다. 이에 관한 자세한 설명은 '퍼스널컴퓨터 1989. 1월호"를 참고하기 바란다.

펑션 10H, 11H, 12H 세개는 각각 펑션 0, 1, 2를 확장시킨 것이라고 생각하면 된다. 요즈음의 프로그램들(예를 들면, 볼랜드의 Side Kick Plus, Quattro, Sprint 등)은 사용중인 BIOS가 확장 키보드 BIOS를 지원하고 있는지 스스로 파악한 다음, 만약 지원하고 있으면 펑션 0, 1, 2 대신 펑션 10H, 11H, 12H를 사용하게 되므로, 행정 전산망용 BIOS 규격을 채택하고 있는 한글 MS-DOS를 사용하는 경우에 한글은 아예 입력되지 않는 현상이 나타난다. 특히, PC-DOS 4.0이나 MS-DOS 4.0에서는 DOS 수준에서 펑션 10H, 11H, 12H를 사용하므로 그 문제는 더욱 심각하게 된다. 별것 아닌 사실인데도 이것을 모르고서 BIOS 수준이나 OS 수준의 한글처리 프로그램을 작성하려면 상당한 애를 먹는 수가 있다. 그 대표적인 경우가 바로 최철용씨의 "한글 도깨비"란 프로그램이다.

리스트 1은 다음과 같은 4개의 모듈로 구성되어 있다.

hio.h

hio.c에서 사용하는 상수, 변수를 정의

hio.c

본 프로그램 모듈로서 다음과 같이 두 개의 함수가 있다.

• word get_u_ch() : 이 함수는 각 키가 눌릴 때마다 코드를 리턴하는 함수로서, 이때 리턴되는 코드는 앞에서 설명한 행정전산망용 BIOS 규격의 키보드 인터럽트(INT 16H)의 펑션 0(AH=F0H)와 같다.

• word get_c_ch() : 이 함수는 행정전산망용 BIOS 규격의 키보드 인터럽트의 펑션 F0(AH=F0H)와 같이 한글의 경우 미완성된 코드는 리턴시키지 않고 완성된 코드만 리턴한다.

usr_fn.c

이 모듈은 사용자가 임의대로 다시 작성할 수 있는 함수들을 따로 모은 것이다. 즉, 사용자는 여기에 있는 함수들을 같은 기능을 수행하는 범위에서 BIOS를 사용하지 않게 한다든지 또는 어셈블리어 프로그램으로 바꾸든지 할 수 있다. 여기에 있는 각 함수들의 기능은 다음과 같다.

• int USR_GETCH() : 이 함수는 원래의 키보드 바이오스(INT 16H)의 펑션 0(AH=0) 또는 10H (AH=10H)와 같은 기능을 수행한다.

• int CAPS_LOCK() : 이 함수는 Caps Lock 상태가 ON이면 0이 아닌 값을 리턴하고 OFF이면 0을 리턴한다.

• int ALT_PRESSED() : 이 함수는 현재 Alt키가 눌린 상태이면 0이 아닌 값을 리턴하고 아니면 0을 리턴한다.

• void USR_PUT_HANGUL(word han_ch) : 한글 한 자를 화면에 표시하는 함수로서 커서는 움직이지 않는다. 한글 미완성 글자를 화면에 표시하

〔표 2〕키보드에 관한 BIOS의 기능 일람(INT 16H)

AH	기 능
00H	keyboard read
01H	keystroke status
02H	shift status
03H	set typematic rate
04H	keyboard click adjustment
05H	keyboard write
06H~0FH	reserved
10H	extended keyboard read
11H	extended keystroke status
12H	extended shift status
13H~FFH	reserved

기 위한 함수로서, 행정전산망용 BIOS 규격의 비디오 인터럽트(INT 10H)의 펑션 FE(AH=FEH)와 비슷한 기능을 수행하나 전달받는 인수가 다르다.

• void USR_DEL_HANGUL( ) : 완성된 한글코드를 리턴하기 전에 지금까지 화면에 표시해 주었던 미완성 한글을 지우기 위한 함수이다. 이렇게 하지 않는 경우에는 NKP를 사용할 때에 보는 것처럼, 이따금 커서위치에 입력하던 글자가 남게되는 현상이 나타난다.

test2kbd.c

이 모듈은 hio.c 모듈이 제대로 동작하는지 테스트하기 위한 것으로서, 오토메타가 어떻게 동작하는지를 파악할 수 있게 해준다.

• 공병우 3벌식 컴퓨터 자판용 한글 입력 오토메타

① 한글코드는 상용 조합형을 사용하기로 한다. 한글 입력자판은 공병우 컴퓨터용 3벌식(**그림 7** 참조)을 사용하기로 한다. 한/영변환 키는 Alt키와 스페이스(Space)키를 동시에 누르는 것으로 한다. 두개의 키를 조합했기 때문에 하나의 키보다는 다소 불편하겠지만, 어느 자판에서나 사용가능하고 비교적 누르기 편리한 위치에 있을 뿐만 아니라 원래 키의 기능을 그대로 사용할 수 있다는 장점이 있으므로 한/영 변환키로서는 무난하리라고 생각한다.

② 입력가능한 자모의 범위는 모아쓴 한글, 단자음, 쌍자음, 복자음, 단모음, 중모음으로 정한다. 초성없이 모음과 받침만으로 구성된 글자나 모음없이 초성과 종성으로만 이루어진 글자는 우리 문자생활에서 전혀 사용되지 않으며 한글의 제작원리에 비추어 보아도 적합치 않으므로 여기에서는 제외하기로 한다.

③ 'BS'키에 대한 처리는 2벌식에서와 같은 방법으로 처리한다.

④ 복자음이나 쌍자음으로 된 받침을 입력하는데 있어서 복자음 또는 쌍자음 키로 입력하는 방법외에 단일 받침의 조합도 허용할 것인가 하는 문제가 있는데, 이를 허용할 경우 오토메타가 다소 복잡해지지만 사용자의 편의와 공병우 3벌식 타자기와의 호환성을 위하여 허용하도록 한다. 단, 이 경우에 있어서도 받침만으로 이루어진 글자를 입력하는데 있어서는 단일 받침의 조합은 허용하지 않고 복자음키만을 사용해서 입력하도록 한다. 이는 좀처럼 사용되지 않는 받침만의 글자를 위하여 전체 오토메타의 흐름을 크게 해치고 싶지 않기 때문이다. 받침 쌍자음에 대하여는 리스트 2의 예제 프로그램에서는 처리하도록 하였으나 **그림 8**의 오토메타의 상태전이도에서는 포함시키지 않았다.

⑤ 한글 입력은 Caps Lock 상태와는 무관하게 이루어진다.

⑥ 한글 미완성 글자에 대한 코드도 돌려줄 수 있도록 한다.

• 프로그램설명

리스트 2(a)의 hiokbw.c는 지금까지 설명한 공병우 3벌식 컴퓨터자판을 위한 한글 오토메타를 구현한 것이다.

여기서 눈여겨 봐야 할 부분은 MYGETCH( )라는 함수이다. MYGETCH( )는 오토메타로 키입력을 제공하는 부분인데 여기서 한글일 경우 CAPS LOCK을 무시하는 작용과 NUM LOCK이 눌러있을 경우 숫자판의 입력을 받을 수 있도록 한다.

〔그림 7〕 공병우 3벌식 컴퓨터용 자판배열

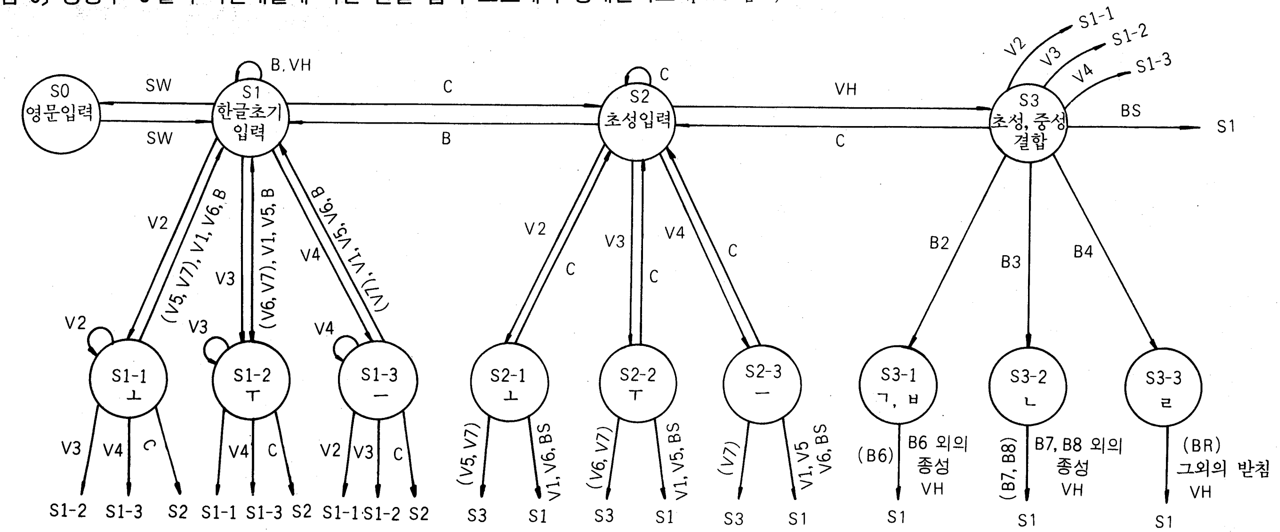
`	1	2	3	4	5	6	7	8	9	0	-	=	\
ㄳ	ㅋ	ㅈ	ㄵ	ㄻ	ㄾ	ㄿ	7	8	9	*	_	+	;
ㄽ	ㅂ	ㅆ	ㅛ	ㅖ	ㅠ	ㅑ	ㄲ	ㅢ	ㄸ	ㅉ	(	)	:

q	w	e	r	t	y	u	i	o	p	[	]
ㅍ	ㅌ	ㅂ	ㅎ	ㅀ	'	4	5	6	/	%	!
ㅅ	ㄹ	ㅕ	ㅣ	ㅓ	ㄹ	ㄱ	ㅁ	ㅊ	ㅍ	ㅋ	?

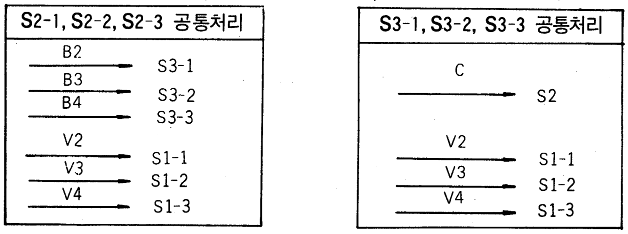
a	s	d	f	g	h	j	k	i	;	'
ㄷ	ㄶ	ㅒ	ㄲ	ㄺ	"	1	2	3	ㅃ	-
ㅇ	ㄴ	ㅐ	ㅏ	ㅡ	ㄴ	ㅇ	ㄷ	ㅈ	ㅂ	ㅌ

z	x	c	v	b	n	m	,	.	/
ㅊ	ㅄ	"	"	ㄼ	ㅆ	ㅇ	<	>	ㅜ
ㅁ	ㄱ	ㅔ	ㅗ	ㅜ	ㅅ	ㅎ	,	.	ㅗ

〔그림 8〕 공병우 3벌식 자판배열에 따른 한글 입력 오토메타 상태전이도 (p171 참조)



- SW가 입력된 경우 S0상태로 간다(S0의 경우는 S1으로)
- C= {모든 초성 자음} ㅗ, ㅜ, ㅡ
  VH= {V1, V5, V6, V7} 오, 우, 으 제외한 모든 모음
  BS= {B1, B5, B6, B7, B8}ㄱ, ㅂ, ㄴ, ㄹ을 제외한 모든 받침
  BR= {B2, B5, B6, B8}ㄱ, ㅂ, ㄴ, ㄹ 뒤에 붙어 복자음을 이룰 수 있는 모든 받침
- α의 입력은 항상 S1의 상태로(S0 제외)
- 괄호는 복자음, 복모음을 구성하는 것을 말함



본 프로그램을 사용하기 위하여는 hiokbw.c를 컴파일하여 오브젝트화일을 만들어 이용프로그램과 링크를 하든지 간단히 #include문을 사용하여 이용프로그램에 삽입하면 된다. 그 간단한 예는 리스트 2(b)의 test3kbd.c를 참조하기 바란다.

한글의 출력에 관하여

한글처리를 하는데 있어서 입력에 관한 연구와 더불어 출력에 관하여도 많은 연구가 필요하다. 여기에서는 한글을 출력하는데 있어서 관계되는 여러가지 사항들에 관하여 알아보기로 한다.

(1) 풀어쓰기와 모아쓰기

한글을 우리가 볼 수 있도록 표시하는 방법에 관한 문제로서 풀어쓰기란 문자 그대로 한글 자모를 결합시키지 않고 풀어서 나열하는 방식을 의미한다. 요즈음은 거의 모든 컴퓨터에서 모아쓰기를 하고 있지만, 모아쓰기를 하는 방식에 대한 연구가 아직 이루어지지 않았던 시절에는 컴퓨터에서 한글을 사용하려면 풀어쓰기를 해야만 가능하고, 아예 이 기회에 풀어쓰기를 바꾸어서 로마자처럼 표시하자고 주장하는 학자들도 있었다.

풀어쓰기 방식은 컴퓨터에 구현하기가 용이해서 초기의 대형이나 미니급 컴퓨터에서 7비트-N바이트 한글코드와 결합하여 많이 사용되었다. 하지만 이후에 한글에 관한 연구에 힘입어 한글 모아쓰기가 가능해졌을 뿐만 아니라 특히 16비트 조합형 코드체계의 등장에 힘입어, 내부 기억장소와 출력이 일치된다는 장점으로 널리 보편화되어 오늘에 이른 것이다.

하지만 한글 풀어쓰기가 우리 컴퓨터 문화에 끼친 영향은 상당히 크다. 특히, 2벌식 글자판 배열은 풀어쓰기에 아주 적합하다는 점에 비추어 볼 때에, 그 제정에 있어서 풀어쓰기 주장자들이 상당한 영향을 끼쳤다는 사실을 무시할 수 없다.

한글에 관하여는 누가 가장 잘 아는가? 당연히 이를 만드신 이가 가장 잘 알 것이다. 한글은 모아쓰는데에 그 장점이 있으며, 한글의 자모 하나하나가 모아쓰기를 하는데 적합하도록 고안되어 있는데 이를 풀어써야 한다고 주장하는 사람은 아예 한글을 버리고 로마자의 발음기호로 대신하자고 주장하지 않는데에 대해서 오히려 감사를 드려야 할 것이다(표 3 참조).

필자가 KOEX에서 보았다는 Eten 한글카드는 한글을 입력하는 과정에서 모아쓰기를 하지않고 풀어쓰기를 하고 있었다. 그네들 나름대로 이유가 있었겠지만, 중요한 사실은 그네들이 한글의 특성을 잘 모르고 한글을 마치 한자처럼 취급하려는데 그 원인이 있는 것이다. 한글은 다른 문자와는 다

른 특성과 장점이 있는 것이다. 이를 잘 살려야만 우리의 한글문화가 더욱 발전할 수 있다는 것은 당연하지 않을까?

(2) 조합형 자형과 완성형 자형

자형(Font)이라고 하는 것은 글자의 모양 즉 글자꼴을 말하는데, 한글의 모양을 처리수준에서 어떻게 처리할 것인지에 관한 문제이다. 조합형 자형이라 함은 한글의 초성, 중성, 종성의 자형을 각기 따로 만든 다음, 글자를 표시하는 과정에서 이를 결합(조합)시켜서 표시하는 것을 말하는데 이를 채택한 방식으로는 NKP, THP와 같은 소프트웨어와 삼보 조합형 한글카드, 옴니테크 조합형 한글카드, 현대 조합형 한글카드 등이 있다. 완성형 자형은 개개의 글자(음절)의 모양을 모두 만들어 두고 해당 글자를 출력할 때에 바로 그 글자의 모양을 표시해주는 것을 말하며, 이를 채택하고 있는 소프트웨어로는 K-System이 있다. 하드웨어로는 대부분의 행정전산망용 한글카드가 이 방식을 채택하고 있다.

〔표 3〕 한글 자모 코드체계

행＼열	0	1	2	3	4	5	6	7	행＼열	12	13	14	15
0					(채움)	ㅀ			0	(채움)	ㅀ		
1					ㄱ	ㅁ			1	ㄱ	ㅁ		
2					ㄲ	ㅂ	ㅏ	ㅛ	2	ㄲ	ㅂ	ㅏ	ㅛ
3					ㄳ	ㅃ	ㅐ	ㅜ	3	ㄳ	ㅃ	ㅐ	ㅜ
4	기				ㄴ	ㅄ	ㅑ	ㅝ	4	ㄴ	ㅄ	ㅑ	ㅝ
5	능				ㄵ	ㅅ	ㅒ	ㅞ	5	ㄵ	ㅅ	ㅒ	ㅞ
6	문				ㄶ	ㅆ	ㅓ	ㅟ	6	ㄶ	ㅆ	ㅓ	ㅟ
7	자				ㄷ	ㅇ	ㅔ	ㅠ	7	ㄷ	ㅇ	ㅔ	ㅠ
8					ㄸ	ㅈ			8	ㄸ	ㅈ		
9					ㄹ	ㅉ			9	ㄹ	ㅉ		
10					ㄺ	ㅊ	ㅕ	ㅡ	10	ㄺ	ㅊ	ㅕ	ㅡ
11					ㄻ	ㅋ	ㅖ	ㅢ	11	ㄻ	ㅋ	ㅖ	ㅢ
12					ㄼ	ㅌ	ㅗ	ㅣ	12	ㄼ	ㅌ	ㅗ	ㅣ
13					ㄽ	ㅍ	ㅘ		13	ㄽ	ㅍ	ㅘ	
14					ㄾ	ㅎ	ㅙ		14	ㄾ	ㅎ	ㅙ	
15					ㄿ		ㅚ		14	ㄿ		ㅚ	

(a) 7단위 한글 자모용 부호 (b) 8단위 한글 자모용 부호

조합형 자형은 조합형 한글코드에 적합한 구조로 다음과 같은 장단점이 있다.

① 각 자모의 자형만 갖추면 되기 때문에 자형을 제작하는데에 시간, 노력, 경비를 절약할 수 있다.
② 자형이 차지하는 용량이 적으므로 메모리를 절약할 수 있으므로, 소프트웨어에 있어서는 사용자 메모리를 적게 차지하며, 하드웨어에 있어서는 롬(ROM)을 적게 사용해도 되므로 비용을 절감할 수 있다.
③ 모든 한글을 다 표시할 수 있다.
④ 포스트스크립을 구현하는데 있어서 완성형에 비하여 훨씬 용이하다.
⑤ 소프트웨어에 있어서 한글을 표시할 때에 자모의 자형을 조합하여 글자의 모양을 만들어 주어야 하므로 이미 다 완성되어 있는 자형을 사용하는 완성형 자형에 비해 출력속도면에서 다소 떨어진다.

반면 완성형 자형은 완성형 한글코드나 한자와 같은 문자에 적합한 구조로서, 조합형 자형의 장단점이 완성형 자형에서는 그 반대로 적용된다.

(3) 비트맵(Bit Map)방식과 포스트스크립(Post Script) 방식

이는 자형을 데이터 구조상 어떻게 처리할 것인지에 관한 문제이다.

비트맵(Bit Map) 방식이란 현재의 거의 모든 한글처리 프로그램에서 사용하고 있는 방식으로 가로, 세로 일정한 규격의 사각형안에서 글자를 점단위로 구성하여 이를 출력하는 방식이다. 현재 NKP나 대부분의 한글카드에서는 한글을 가로와 세로 모두 16개의 점으로 구성된 자형을 사용하고 있다. 모든 형체를 세분하면 점으로 이루어지므로 어떠한 글자이든지 비교적 쉬운 방법으로 표현할 수 있다는 장점이 있지만, 글자를 확대 또는 축소할 경우 그 배율이 크면 클수록 글자의 모양이 떨어진다는 문제점을 지니고 있다.

이에 비하여 포스트스크립(Post Script)방식은 글자의 구성요소를 점으로 보는 것이 아니라, 길이와 각도를 가진 벡터로 취급하는 것이다. 따라서 이 방식은 비트맵 방식에 비하여 자형을 제작하기

가 어려운대신 확대, 축소가 비트맵에 비하여 훨씬 자유로우며 또한 글자 모양을 원형에 가깝게 유지할 수가 있다.

한글의 경우에는 오토캐드에서나 이러한 방식을 구경할 수 있는데, 아직 워드프로세서 같은데에서 포스트스크립 방식의 자형을 사용하는 것을 보지 못했다. 앞으로 DTP(Desk Top Publishing : 탁상출판)를 지향하는 소프트웨어에서는 이 방식이 필수적으로 채택되어야 할 것 같은데 그러기 위해서는 조합형 코드체계와 조합형 자형구조를 채택하는 것이 바람직할 것이다.

(4) 한글카드에 관하여

현재 우리 주위에는 많은 한글카드가 나돌고 있다. 이미 VGA 컬러그래픽 보드에서도 한글을 사용할 수 있는 카드도 나오고 있긴 하지만 이들중 과연 몇 퍼센트가 우리 한글의 특징을 잘 파악하고 그에 따라 처리를 하고 있는지는 의문이다. 한글카드에서 하는 일은 미리 지정된 코드구조에 따라 비디오 버퍼(Video Buffer : 화면에 표시되는 메모리 영역)를 탐색해가며 영문인지 한글코드인지 구분한 다음 한글코드에 해당되면 문자롬에 정의되어 있는 한글자형을 화면에 보여주는 일을 수행한다. 한글카드를 구입하기 전에 알아두어야 할 일은 다음과 같다.

① 어느 한글코드 체계를 사용하고 있는 것인가? 여러가지 한글코드를 지원하는 경우 기본이 되는 한글코드 체계는 어느 것인가?

요즈음 나오는 한글카드는 대개 다음과 같은 세가지 유형으로 나눌 수 있다.

a. 상용 조합형 : 삼보 조합형, 옴니테크 KS 한글카드, 현재 조합형 한글카드 등이 있다.

b. 표준 완성형 : 행정전산망을 선전문구로 운용하는 대개의 한글카드는 이 코드체계를 채택하고 있다.

c. 여러가지 코드 지원 : 대우의 KLDC 한글카드가 대표적인 경우이며 상용조합형, 표준완성형, 그리고 DH 한글코드 이렇게 세가지를 지원하고 있다. 하지만 이렇게 여러가지 코드체계를 지원하고 있다고 해서 다 좋은 것은 아니다. 중요한 것은 어느 코드체계를 기본으로 채택하고 있는가를 알아보아야 할 것이다. 대개 조합형 코드를 기본체계로 하든지 아니면 표준완성형을 기본체계로 하는 것이 보통인데, 대우 KLDC 한글카드의 경우에는 표준완성형을 그 기본으로 채택하고 있다. 조합형코드를 기본으로 채택하고 있다면 조합형코드를 사용할 때만큼은 모든 한글을 다 입력할 수 있어야 한다. 그런데 조합형코드를 사용할 때에도 2,350자 밖에 입력할 수 없다면 이것은 표준완성형을 기본체계로 채택하고 있고 조합형은 코드변환 기법에 의하여 지원하고 있는 것이다.

자신이 현재 사용하고 있는 한글코드와는 다른 코드체계를 사용하고 있는 한글카드를 구입할 경우에는, 이미 입력해 놓은 데이터에 대해서 코드변환을 해야 하는 부담이 있으므로 신중히 고려하여야 할 일이다.

② 글자를 입력할 때에 모아쓰기는 제대로 해주고 있는가? 혹, 입력도중 글자가 사라졌다 다시 나타났다 하는 현상은 없는가?

이 문제는 모아쓰기 오토메타를 잘 구현하고 있느냐에 관한 것으로서, 앞서 언급한 Eten 한글카드와 같은 경우에 있어서는 입력시 풀어쓰기를 하고 있는 전근대적인 방식을 채택하고 있으며, K-System의 경우에는 표준완성형 한글코드를 채택하고 있는데 갑자기 입력중인 글자가 화면에서 사라지는 현상이 나타난다. 이는 오토메타의 문제로서 완성형코드의 오토메타는 조합형과는 달리 각 상태가 종료될 때마다 지금까지 만들어진 코드가 완성형코드에 속하는지 검사해 주는 과정이 필요한데 이 과정이 결여되어 있기 때문이다.

③ 영문 소프트웨어에서 문제없이 잘 동작하는가?

삼보 조합형 한글카드는 그래픽과 텍스트를 하나의 화면에서 동시에 표시할 수 있는 장점이 있으나, 몇몇 영문 소프트웨어를 사용하다 보면 화면이 갑자기 사라지는 문제가 발생한다. 이는 허큘리스 그래픽 카드에서 스크린의 전환을 제어하기 위하여 사용하는 3B8H번지 포트(Port)의 특정 비트를 한글화면과 영문화면의 선택에 사용해서이다.

④ 이미 사용하고 있는 모니터와 연결해서 사용할 수 있는가?

새로 모니터까지 구입할거라면 별로 문제가 안되지만, 이미 사용하고 있는 모니터를 그대로 사용하려면 한번쯤 짚고 넘어가야 할 것이다. 요즈음 24×24 자형을 사용하는 한글카드도 나오고 있는데 이는 16×16 자형에 비하여 글자의 모양이 훨씬 좋은 점이 있기는 하지만, 기존의 모니터를 사용할 수 없거나 또는 사용가능하더라도 화면이 떨리는 현상이 발생한다. 또는 대우 KLDC 한글카드의 경우에는 보통의 흑백 모니터에 사용하면 화면의 떨림 때문에 더 높은 주파수를 지원하는 모니터를 사용하여야 한다.

맺음말

지금까지 여러가지를 다룬듯하지만 한글분야는

▨ 특집 한글 입출력에 관한 연구

그 범위가 매우 넓고 필자의 능력밖의 것도 많아서 여기에서 다루지 못한 것들도 많다. 한글 포스트스크립, 한글 철자검사뿐만 아니라 한글 문자인식등 해야할 것들은 많은데 우리의 현실을 돌이켜보노라면 갈길은 멀고도 험난하기도 하다.

영문은 타자하는 대신 스캐너라고 하는 입력장치를 통해 책의 내용을 복사하거나 훑으며 지나가듯이 입력할 수 있는데, 우리는 언제까지 눈으로 보아가면서 키보드로 입력해야 할까? 신속하게 변화하는 정보시대에 우리는 언제까지 코드문제, 자판문제에만 매달려야 할까? 앞으로 남북한이 통일될 때를 대비하여 미리 한글코드나 자판의 통일에 대하여 양측 관계자들이 협의를 가져보는 것도 바람직하지 않을까? 아직 남한만의 통일도 이루지 못한터에 너무 욕심을 부린 말인지도 모르겠다.

글을 마칠때면 하던 일을 끝냈다는 어떤 후련한 느낌과 한편으로는 자신이 한 일에 대하여 잘못된 부분은 없는지하는 염려가 함께 온다. 지금까지 이 글을 읽어주신 분들께 감사드리며 아무쪼록 한글 분야에 관심이 있는 분들에게 조그마한 도움이 되었으면 하며, 부디 한글의 특성을 살린 좋은 한글 프로그램이 많이 쏟아져 나오길 바라면서 이 글을 마치고자 한다.

〈참고문헌〉

1. 한글 표준안 코드에 대하여, 신홍철, 마이크로소프트웨어 1987. 11
2. 행정전산망용 다기능사무기기 종합보고서, 한국데이타통신 주식회사, 1988. 7
3. 한글 정보처리 표준화 연구, 한국표준 연구소, 1988.7
4. 한글, 한자 코드 표준화에 관한 연구, 한국표준연구소, 1987. 2
5. 군사문화가 짓밟은 한글 기계화 정책, 송현(한글 기계화 추진회 회장), 1988. 12

〔리스트 1〕 2벌식 한글 입력 오토메타의 상태전이도에 따른 예제 프로그램

```
/*****************************************************************************
   module name : hio.h
*****************************************************************************/

/*-------------------------------------------------------------------------*/
/*   global constants                                                      */
/*-------------------------------------------------------------------------*/
enum { ALPHA, C1, C2, C3, C4, C5, C6, C7, C8,
      C9, V1, V2, V3, V4, V5, V6, V7 };   /* character class              */

enum { C_FILL=1, C_GI, C_GI2, C_NI, C_DI, C_DI2, C_RI,
      C_MI, C_BI, C_BI2, C_SI, C_SI2, C_YI, C_JI,
      C_JI2, C_CHI, C_KI, C_TI, C_PI, C_HI };
                              /* consonant (5 bit)
enum { V_FILL=2, V_AH, V_AE, V_YA, V_YAE, V_UH, V_EH=10, V_YU, V_YE,
      V_OH, V_WA, V_WAE, V_WE=18, V_YO, V_OO, V_WO, V_WYE,
      V_WI, V_YOO=26, V_EU, V_EUI, V_YI };
                              /* basic vowel (5 bit)                       */

enum { B_FILL=1, B_GI, B_GI2, B_GS, B_NI, B_NJ, B_NH, B_DI, B_RI, B_RG,
      B_RM, B_RB, B_RS, B_RT, B_RP, B_RH, B_MI, B_BI=19, B_BS,
      B_SI, B_SI2, B_YI, B_JI, B_CHI, B_KI, B_TI, B_PI, B_HI };
                              /* basic bottom-consonant                    */

#define U_HANGUL  0xF0  /* extended code for uncompleted Hangul            */
#define C_HANGUL  0xF1  /* extended code for completed Hangul              */

#define MAX_RETURN_QUEUE 5

/*-------------------------------------------------------------------------*/
/*   variable type                                                         */
/*-------------------------------------------------------------------------*/
typedef unsigned char byte;
typedef unsigned word;

typedef struct {
   unsigned third  : 5;
   unsigned second : 5;
   unsigned first  : 5;
   unsigned flag   : 1;
} han2byte;

typedef union {
   han2byte n3;
   word     n2;
} hancode;

/*-------------------------------------------------------------------------*/
/*   global variables                                                      */
/*-------------------------------------------------------------------------*/
static word return_queue [MAX_RETURN_QUEUE];
static int
   return_queue_head = 0,
   return_queue_tail = 0;

/*-------------------------------------------------------------------------*/
/*   status definition                                                     */
/*-------------------------------------------------------------------------*/
#define S0     0    /* English mode                                        */
#define S1     1    /* begin Hangul input                                  */

#define S1_1   2    /* vowel(o)                                            */
#define S1_2   3    /* vowel(u)                                            */
#define S1_3   4    /* vowel(eu)                                           */
#define S1_C   5    /* consonant(giyuk,bieup,nieun,rieul)                  */
#define S2     6    /* consonant                                           */
```

```
#define S2_1   7    /* consonant + vowel(o)                              */
#define S2_2   8    /* consonant + vowel(u)                              */
#define S2_3   9    /* consonant + vowel(eu)                             */
#define S3     10   /* consonant + vowel                                 */
#define S3_1   11   /* consonant + vowel + consonant(giyuk,bieup)        */
#define S3_2   12   /* consonant + vowel + consonant(nieun)              */
#define S3_3   13   /* consonant + vowel + consonant(rieul)              */
#define S4     14   /* consonant + vowel + consonant                     */
```

```
/*------------------------------------------------------------------------*/
/*  key definition                                                        */
/*------------------------------------------------------------------------*/
#define BS     0x0E08  /* Back Space to cancel last key-in                 */
#define SW     0x3920  /* Space key for Hangul/Enlish switching with Alt
                          key combination                                  */
```

(a) hio.h

```
/***************************************************************************/
/*                                                                         */
/*      Module name : HIO.C (Hangul input automata)                        */
/*          Version : V 0.02                                               */
/*         Function : analyze keyboard input stream and generate           */
/*                    2-byte popular combination code                      */
/*                                                                         */
/*               OS : PC-DOS 4.0                                           */
/*         Compiler : Turbo C  Version 2.0  (S model)                      */
/*        Processor : Intel 8088/8086/80286                                */
/*                                                                         */
/*    Function Usage :                                                     */
/*    --------------                                                       */
/*                                                                         */
/*      word get_u_ch() : return code when each key stroke                 */
/*      word get_c_ch() : return only completed Hangul, the others same    */
/*                        as get_u_ch()                                    */
/*                                                                         */
/*       * Hangul/English switching key : Alt-Space combination            */
/*                                                                         */
/*   Log for changes :                                                     */
/*   -----------------                                                     */
/*   88.12.09 KISS V0.01 - move the first hands on the keyboard for this   */
/*                         program.                                        */
/*   89.12.11 KISS V0.01 - finish coding, testing and debugging            */
/*   89. 7. 5 KISS V0.02                                                   */
/*        . combined-vowels without consonants and compound-consonants     */
/*          occupying bottom position are included in user input           */
/*          catagory                                                       */
/*        . reform each status No. and 4 status(S1-1,S1-2,S1-3,S1-C)       */
/*          appended, the previous 3 for combined-vowels handling and      */
/*          the last one for consonant clusters                            */
/*        . separate get_c_ch() from get_u_ch()                            */
/*                                                                         */
/***************************************************************************/

/*------------------------------------------------------------------------*/
/*  header files                                                          */
/*------------------------------------------------------------------------*/
#include <ctype.h>
#include <string.h>
/*------------------------------------------------------------------------*/
/*  user function                                                         */
/*------------------------------------------------------------------------*/

#include "usr_fn.c"

/*
   word USR_GETCH()  : return ascii/scan code for every each keystoke.

   int CAPS_LOCK()   : return non-zero if Caps-Lock on otherwise zero.

   int ALT_PRESSED() : return non-zero if Alt pressed otherwise zero.

   void USR_PUT_HANGUL(word hangul) : put 2 byte hangul code on the screen
                      and do not move the cursor to the right.

   void USR_DEL_HANGUL() : delete the uncompleted hangul displayed on the
                      current cursor position.
*/

/***************************************************************************/
static word pop_queue (void)
{
   word temp;

   if (return_queue_head == return_queue_tail)
      return NULL;

   else  {
      temp = return_queue[return_queue_head];
      if (++return_queue_head >= MAX_RETURN_QUEUE)
         return_queue_head = 0;
      return temp;
   }
}

/***************************************************************************/
static void push_queue (word code)
{
   return_queue[return_queue_tail] = code;
   if (++return_queue_tail >= MAX_RETURN_QUEUE)
      return_queue_tail = 0;
}

/***************************************************************************/
static word combine2 (word code1, word code2, word code3)
{
   hancode code;

   code.n3.flag   = 1;
   code.n3.first  = code1;
   code.n3.second = code2;
   code.n3.third  = code3;
   return code.n2;
}

/***************************************************************************/
static int char_class (int ch)
{
   char *cp, s[] = "EQWReTdczrqsfaxvtwgiOuPybhnmkojpl";
   int  class[] = { C1,C1,C1,C2,C2,C2,C2,C2,C2,C3,C3,C4,C5,C6,C6,C6,
      C7,C8,C9,V1,V1,V1,V1,V1,V1,V2,V3,V4,V5,V5,V6,V6,V7 };

   if ((cp=strchr(s, ch)) == NULL) return ALPHA;
   else return class[cp-s];
}

/***************************************************************************/
static int to_c5bit (int ch)
{
   char s[] = "rRseEfaqQtTdwWczxvg";
   int  v5bit_tbl[] = { V_AH, V_AE, V_YA, V_YAE, V_UH, V_EH,
      V_YU, V_YE, V_OH, V_YO, V_OO, V_YOO, V_EU, V_YI };

   return v5bit_tbl[strchr(s,ch)-s];
```

```
}

/*************************************************************************/
static int to_b5bit (int ch)
{
   char s[] = "rRsefaqtTdwczxvg";
   int  b5bit_tbl[] = { B_GI, B_GI2, B_NI, B_DI, B_RI, B_MI, B_BI,
      B_SI, B_SI2, B_YI, B_JI, B_CHI, B_KI, B_TI, B_PI, B_HI };

   return b5bit_tbl[strchr(s,ch)-s];
}

/*************************************************************************/
static int rieul_complex (int ch)
{
   char *sp, s[] = "raqtxvg";
   int  rc[] = { B_RG, B_RM, B_RB, B_RS, B_RT, B_RP, B_RH };

   if ((sp=strchr(s,ch)) == NULL)
      return NULL;
   else return rc[sp-s];
}


/*************************************************************************/
#define MAX_CODE_QUEUE 6

word get_u_ch (void)
{
   word temp, ch;
   int  class;

   static word last_c, last_v, last_b;

   static struct  {
      hancode code;
      int  station;
   } code_queue [MAX_CODE_QUEUE] =  {
      {{B_FILL, V_FILL, C_FILL, 1}, S1} };

   static int
      code_queue_ptr = 0,
      status         = S0;


   if ((temp=pop_queue()) != NULL) return temp;

   loop: /*------------------------------ loop start here -------------------*/
   ch = USR_GETCH();

   if (ch == SW && ALT_PRESSED())  {      /* SW key handling                 */
      if (status == S0)  {
         status = S1;
         goto loop;
      }
   else if (status == S1)  {
      status = S0;
      goto loop;
   }
   else  {
      status = S0;
      _DX = code_queue[code_queue_ptr].code.n2;
      code_queue_ptr = 0;
      _AL = _DL;
      _AH = C_HANGUL;
      push_queue(_AX);
      _AL = _DH;
      _AH = C_HANGUL;
      return _AX;
   }
}

if (status == S0) return ch;  /* status S0 handling                         */

if (ch == BS)  {              /* 'BS' key handling                          */
   if (status == S1) return ch;
   else  {
      --code_queue_ptr;
      status = code_queue[code_queue_ptr].station;
      _DX = code_queue[code_queue_ptr].code.n2;
      _AL = _DL;
      _AH = U_HANGUL;
      push_queue(_AX);
      _AL = _DH;
      _AH = U_HANGUL;
      return _AX;
   }
}

if (strchr("YUIASDFGHJKLZXCVBNM", ch) != NULL)  /* ignore upper case        */
   ch = ch ^ 0x0020;
if (CAPS_LOCK() && (strchr("qQwWeErRtToOpP", ch) != NULL))
   ch = ch ^ 0x0020;                             /* ignore CAPS LOCK         */

if ((class=char_class(ch)) == ALPHA)  {         /* 'Alpha' handling         */
   if (status==S1) return ch;
   else  {
      _DX = code_queue[code_queue_ptr].code.n2;
      status = S1;
      code_queue_ptr = 0;      /* initialize code queue                     */

      _AL = _DL;
      _AH = C_HANGUL;
      push_queue(_AX);
      push_queue(ch);
      _AL = _DH;
      _AH = C_HANGUL;
      return _AX;
   }
}

switch (status)  {

   case  S1 :
   if (class <= C9)  {      /* consonant                                   */
      code_queue_ptr ++;
      status = code_queue[code_queue_ptr].station
             = (class==C3 || class==C4 || class == C5) ? S1_C : S2;

   last_c = to_c5bit(ch);
   _DX = code_queue[code_queue_ptr].code.n2
      = combine2(last_c, V_FILL, B_FILL);

   _AL = _DL;
   _AH = U_HANGUL;
```

```
   push_queue(_AX);

   _AL = _DH;
   _AH = U_HANGUL;
   return _AX;  }
else  {                      /* vowel except V2, V3, V4                    */
   if (class != V2 && class != V3 && class != V4)  {
      code_queue_ptr = 0;
      status = S1;
      _DX = combine2(C_FILL, to_v5bit(ch), B_FILL);
      _AL = _DL;
      _AH = C_HANGUL;
      push_queue(_AX);

      _AL = _DH;
      _AH = C_HANGUL;
      return _AX;
   }
   else  {               /* V2, V3, V4                                  */
      code_queue_ptr ++;
      switch (class)  {
         case V2 : status = S1_1;  break;
         case V3 : status = S1_2;  break;
         case V4 : status = S1_3;
      }

      last_v = to_v5bit(ch);
      _DX = code_queue[code_queue_ptr].code.n2
         = combine2(C_FILL, last_v, B_FILL);

      _AL = _DL;
      _AH = U_HANGUL;
      push_queue(_AX);

      _AL = _DH;
      _AH = U_HANGUL;
      return _AX;
   }
   }

case  S1_1 :

   if (class <= C9)        /* consonant                                   */
      goto label_S2;

   switch (last_v=to_v5bit(ch))  {
      case  V_AH : temp = V_WA;   break;
      case  V_AE : temp = V_WAE;  break;
      case  V_YI : temp = V_WE;  break;
      default     : goto OTHER_VOWELS;
      }

COMBINED_VOWELS:
      code_queue_ptr = 0;
      status = S1;
      _DX = combine2(C_FILL, temp, B_FILL);
      _AL = _DL;
      _AH = C_HANGUL;
      push_queue(_AX);
      _AL = _DH;
      _AH = C_HANGUL;
      return _AX;

OTHER_VOWELS:
      _DX = code_queue[code_queue_ptr].code.n2;
      _AL = _DH;
      _AH = C_HANGUL;
      push_queue(_AX);
      _AL = _DL;
```

```
   _AH = C_HANGUL;
   push_queue(_AX);

   switch (class)  {
      case V2 : status = S1_1;  break;
      case V3 : status = S1_2;  break;
      case V4 : status = S1_3;  break;
      default : status = S1;
   }

   if (status == S1)   {
      code_queue_ptr = 0;
      _DX = combine2(C_FILL, last_v, B_FILL);
      _AL = _DH;
      _AH = C_HANGUL;
      push_queue(_AX);
      _AL = _DL;
      _AH = C_HANGUL;
      push_queue(_AX);
      return pop_queue();
   }
   else  {
      code_queue_ptr = 1;
      _DX = code_queue[code_queue_ptr].code.n2
         = combine2(C_FILL, last_v, B_FILL);
      _AL = _DH;
      _AH = U_HANGUL;
      push_queue(_AX);
      _AL = _DL;
      _AH = U_HANGUL;
      push_queue(_AX);
      return pop_queue();
   }

case  S1_2 :

   if (class <= C9)        /* consonant                                   */
      goto label_S2;

   switch (last_v=to_v5bit(ch))  {
      case  V_UH : temp = V_WO;   break;
         case  V_EH : temp = V_WYE;  break;
         case  V_YI : temp = V_WI;  break;
         default     : goto OTHER_VOWELS;
      }
      goto COMBINED_VOWELS;
case  S1_3 :

   if (class <= C9)        /* consonant                                   */
      goto label_S2;

   if ((last_v=to_v5bit(ch)) == V_YI)  {
      temp = V_EUI;
      goto COMBINED_VOWELS;
   }
   else goto OTHER_VOWELS;


case  S1_C :
   switch (last_c)  {
      case C_GI : if (class == C7)  {
                     temp = B_GS;  break;
                  }
                  else goto label_S2;
      case C_BI : if (class == C7)  {
                     temp = B_BS;  break;
                  }
                  else goto label_S2;
```

```
    case C_NI : if (class == C8)  {
                  temp = B_NJ;  break;
               }
               else if (class == C9)  {
                  temp = B_NH;  break;
               }
               else goto label_S2;
    case C_RI : if ((temp=rieul_complex(ch)) == NULL)
                  goto label_S2;
}

status = S1;
code_queue_ptr = 0;

_DX = combine2(C_FILL, V_FILL, temp);
_AL = _DL;
_AH = C_HANGUL;
push_queue(_AX);
_AL = _DH;
_AH = C_HANGUL;
return _AX;


     case  S2 :
label_S2:
        if (class <= C9)  {      /* consonant                        */
           _DX = code_queue[code_queue_ptr].code.n2;
           _AL = _DH;
           _AH = C_HANGUL;
           push_queue(_AX);
           _AL = _DL;
           _AH = C_HANGUL;
           push_queue(_AX);

           code_queue_ptr = 1;
           status = code_queue[code_queue_ptr].station
                 = (class==C3 || class==C4 || class == C5) ? S1_C : S2;

           last_c = to_c5bit(ch);
           _DX = code_queue[code_queue_ptr].code.n2
              = combine2(last_c, V_FILL, B_FILL);

           _AL = _DH;
           _AH = U_HANGUL;
           push_queue(_AX);
           _AL = _DL;
           _AH = U_HANGUL;
           push_queue(_AX);

           return pop_queue();
        }
        else  {                  /* vowel                            */
           code_queue_ptr ++;
           switch (class)  {
              case V2 : status =
                 code_queue[code_queue_ptr].station = S2_1;  break;
              case V3 : status =
                 code_queue[code_queue_ptr].station = S2_2;  break;
              case V4 : status =
                 code_queue[code_queue_ptr].station = S2_3;  break;
              default : status =
                 code_queue[code_queue_ptr].station = S3;
           }
           last_v = to_v5bit(ch);

vowel_handle:
           _DX = code_queue[code_queue_ptr].code.n2
             = combine2(last_c, last_v, B_FILL);
           _AL = _DH;
           _AH = U_HANGUL;
           push_queue(_AX);
           _AL = _DL;
           _AH = U_HANGUL;
           push_queue(_AX);
           return pop_queue();
        }


     case  S2_1 :

        if (class!=V5 && class!=V7)  goto label_S3;

        switch (to_v5bit(ch))  {
           case  V_AH : last_v = V_WA;   break;
           case  V_AE : last_v = V_WAE;  break;
           case  V_YI : last_v = V_WE;
        }

vowel_handle2:
        code_queue_ptr ++;
        code_queue[code_queue_ptr].station = S3;
        goto vowel_handle;
     case  S2_2 :

        if (class!=V6 && class!=V7)  goto label_S3;

        switch (to_v5bit(ch))  {
           case  V_UH : last_v = V_WO;   break;
           case  V_EH : last_v = V_WYE;  break;
           case  V_YI : last_v = V_WI;
        }
        goto vowel_handle2;


     case  S2_3 :

        if (class != V7)  goto label_S3;

        last_v = V_EUI;
        goto vowel_handle2;


     case  S3 :
label_S3:
        if (class == C1)         /* consonant                        */
           goto label_S2;

        if (class <= C9)  {
           code_queue_ptr ++;
           switch (class)  {
              case C3 : status =
                 code_queue[code_queue_ptr].station = S3_1;  break;
              case C4 : status =
                 code_queue[code_queue_ptr].station = S3_2;  break;
              case C5 : status =
```

```
            code_queue[code_queue_ptr].station = S3_3; break;
          default : status =
            code_queue[code_queue_ptr].station = S4;
        }
        last_b = ch;
        _DX = code_queue[code_queue_ptr].code.n2
          = combine2(last_c, last_v, to_b5bit(ch));
bottom_handle1:
        _AL = _DL;
        _AH = U_HANGUL;
        push_queue(_AX);
        _AL = _DH;
        _AH = U_HANGUL;
        return _AX;
      }
      else                    /* vowel                                   */
        goto OTHER_VOWELS;
    case  S3_1 :              /* B_GI, B_BI                              */
      if (class != C7)  goto label_S4;

      temp = (to_b5bit(last_b) == B_GI) ? B_GS : B_BS;

  bottom_handle2:
      code_queue_ptr ++;      /* code_queue_ptr = 3 or 4                 */
      status = code_queue[code_queue_ptr].station = S4;
      last_b = ch;
      _DX = code_queue[code_queue_ptr].code.n2
        = combine2(last_c, last_v, temp);
      goto bottom_handle1;

    case  S3_2 :              /* B_NI                                    */
      if (class == C8)  {
        temp = B_NJ;
        goto bottom_handle2;
      }
      else if (class == C9)  {
        temp = B_NH;
        goto bottom_handle2;
      }
      else goto label_S4;


    case  S3_3 :              /* B_RI                                    */

      if ((temp=rieul_complex(ch)) == NULL) goto label_S4;
      else goto bottom_handle2;


    case  S4 :
label_S4:
      if (class <= C9)        /* consonant                               */
        goto label_S2;

      else  {                 /* vowel                                   */
        _DX = code_queue[code_queue_ptr-1].code.n2;
        _AL = _DH;
        _AH = C_HANGUL;
        push_queue(_AX);
        _AL = _DL;
        _AH = C_HANGUL;
        push_queue(_AX);

        code_queue_ptr = 1;
        last_c = to_c5bit(last_b);
        code_queue[code_queue_ptr].code.n2 =
          combine2(last_c, V_FILL, B_FILL);
        goto label_S2;
      }
    }
}


/****************************************************************************/
#define EXIST     1
#define NONEXIST  0

word get_c_ch (void)
{
  word ch;
  static int  second_byte;

  if (second_byte == EXIST)  {
    second_byte = NONEXIST;
    return get_u_ch();
  }
  while(((ch = get_u_ch()) >> 8) == U_HANGUL)
    USR_PUT_HANGUL((ch<<8) | (get_u_ch() & 0x00FF));

  if ((ch)>>8) == C_HANGUL)  {
    USR_DEL_HANGUL();         /* delete previous uncompleted hangul     */
    second_byte = EXIST;      /*    displayed on the screen when        */
    return ch;                /*    return completed hangul code        */
  }
  else return ch;
}
```

(b) hio.c

```
/****************************************************************************
   module name : usr_fn.c

****************************************************************************/

#include <bios.h>
#include <stdio.h>

#define USR_GETCH()      bioskey(0)          /* key input function          */
#define CAPS_LOCK()      (bioskey(2) & 0x40) /* determine if Caps Lock on   */
#define ALT_PRESSED()    (bioskey(2) & 8)    /* determine if Alt pressed    */

void USR_PUT_HANGUL (word han_ch)
{
  putchar(han_ch >> 8);
  putchar(han_ch);
  putchar(BS);
  putchar(BS);
}

void USR_DEL_HANGUL (void)
{
  putchar(' ');
  putchar(' ');
  putchar(BS);
  putchar(BS);
}
```

(c) usr_fn.c

```
/*****************************************************************************
   Moudule name : test2kbd.c

*****************************************************************************/
#include "hio.h"
#include "hio.c"
#include <stdio.h>

/*****************************************************************************/
#define QUIT      0x011B        /* ESC key                                   */

main()
{
   word ch;

   while ((ch=get_c_ch()) != QUIT)  {
      putchar(ch);
   }
}
```

(d) test2kbd.c

〔리스트 2〕 3벌식 한글 입력 오토메타의 상태전이도에 따른 예제 프로그램

```
/*************************************************************************/
/*                                                                       */
/*        module name:  HIOKBW.C                                         */
/*   descriptive name:  Hangul basic Input/Output for Kong ByongWoo      */
/*            version:  Ver 0.1                                          */
/*           function:  This is a simple program to convert keyboard    */
/*                      input stream into 2-byte combination Hangul code */
/*                                                                       */
/*                * Hangul/English switching : ALT-Space combination     */
/*                                                                       */
/*        environment:  PC-DOS 4.0                                       */
/*           compiler:  Quick C  Version 1.0  (S model)                  */
/*          processor:  Intel 8088/8086/80286                            */
/*                                                                       */
/* change activity log:                                                  */
/* -------------------                                                   */
/*     89.7.19 KHT  V0.1   - coding, testing and debugging               */
/*************************************************************************/

/*-----------------------------------------------------------------------*/
/*  header files                                                         */
/*-----------------------------------------------------------------------*/
#include <stdio.h>
#include <bios.h>
#include <ctype.h>
#include <string.h>
#include <dos.h>

/*-----------------------------------------------------------------------*/
/*  global constants                                                     */
/*-----------------------------------------------------------------------*/
enum {  ALPHA, C, V1, V2, V3, V4, V5, V6, V7,
        B1, B2, B3, B4, B5, B6, B7, B8  };        /* character class     */

enum {  C_GI=2, C_GI2, C_NI, C_DI, C_DI2, C_RI,
        C_MI, C_BI, C_BI2, C_SI, C_SI2, C_YI, C_JI,
        C_JI2, C_CHI, C_KI, C_TI, C_PI, C_HI };
                                        /* consonant (5 bit)             */
enum {  V_AH=3, V_AE, V_YA, V_YAE, V_UH, V_EH=10, V_YU, V_YE,
        V_OH, V_WA, V_WAE, V_WE=18, V_YO, V_OO, V_WO, V_WYE,
        V_WI, V_YOO=26, V_EU, V_EUI, V_YI };
                                        /* basic vowel (5 bit)           */

enum {  B_GI=2, B_GI2, B_GS, B_NI, B_NJ, B_NH, B_DI, B_RI, B_RG,
        B_RM, B_RB, B_RS, B_RT, B_RP, B_RH, B_MI, B_BI=19, B_BS,
        B_SI, B_SI2, B_YI, B_JI, B_CHI, B_KI, B_TI, B_PI, B_HI };
                                        /* basic bottom-consonant        */

#define U_HANGUL     0xF0     /* extended code for uncompleted Hangul      */
#define C_HANGUL     0xF1     /* extended code for completed Hangul        */
#define HANJA        0xF2     /* extended code for Hanja                   */

#define MAX_RETURN_QUE 5
#define MAX_CODE_STACK 6

/*-------------------------------------------------------------------------*/
/*   variable type                                                          */
/*-------------------------------------------------------------------------*/
typedef unsigned char byte;
typedef unsigned word;

typedef struct
    {
    unsigned third  : 5;
    unsigned second : 5;
    unsigned first  : 5;
    unsigned flag   : 1;
    }   han2byte;

typedef union
    {
    han2byte n3;
    word     n2;
    } hancode;

struct  WORD_REG
        {
        unsigned int AX;
        unsigned int DX;
        };

struct  BYTE_REG
        {
        unsigned char AL, AH;
        unsigned char DL, DH;
        };

union   USER_REG
        {
        struct  WORD_REG x;
        struct  BYTE_REG h;
        } IORegs;


/*-------------------------------------------------------------------------*/
```

```c
/*  user function                                                          */
/*-------------------------------------------------------------------------*/
#define INT16_0         _bios_keybrd(0)             /* key board read      */

/*  input status                                                           */
#define S0      0       /* English mode                                    */
#define S1      1       /* waiting for beginning consonants                */
#define S2      2       /* waiting for vowel                               */
#define S3      3       /* waiting for consonant clusters                  */

#define S1_1    4       /* vowel(o)                                        */
#define S1_2    5       /* vowel(u)                                        */
#define S1_3    6       /* vowel(eu)                                       */


#define S2_1    7       /* consonant + vowel(o)                            */
#define S2_2    8       /* consonant + vowel(u)                            */
#define S2_3    9       /* consonant + vowel(eu)                           */

#define S3_1    10      /* consonant + vowel + consonant(giyuk,bieup)      */
#define S3_2    11      /* consonant + vowel + consonant(nieun)            */
#define S3_3    12      /* consonant + vowel + consonant(rieul)            */

#define SI      0x8600  /* English Start                                   */
#define SO      0x8500  /* Hangul Start                                    */
#define BS      0x0E08  /* Back Space                                      */
#define SPACE   0x3920  /* Space                                           */
#define C_FILL  1   /* consonant filler                                    */
#define V_FILL  2   /* vowel filler                                        */
#define B_FILL  1   /* bottom-consonant filler                             */
/*-------------------------------------------------------------------------*/
/*  global variables                                                       */
/*-------------------------------------------------------------------------*/
static word return_que [MAX_RETURN_QUE];
static int
    return_que_head = 0,
    return_que_tail = 0;

static word last_c, last_v, last_b;
static struct
    {
    hancode code;
    int  station;
    } code_stack [MAX_CODE_STACK] =
    {{{B_FILL, V_FILL, C_FILL, 1}, S1} };

static int
    code_stack_ptr  = 0,
    status          = S1;

/***************************************************************************/
static word delete_que (void)
    {
    word temp;

    if (return_que_head == return_que_tail)
        return NULL;
    else
        {
        temp = return_que[return_que_head];
        if (++return_que_head >= MAX_RETURN_QUE)
            return_que_head = 0;
        return temp;
        }
    }

/***************************************************************************/
static void add_que (word code)
    {
    return_que[return_que_tail] = code;

    if (++return_que_tail >= MAX_RETURN_QUE)
        return_que_tail = 0;
    }

/***************************************************************************/
static word combine2 (word code1, word code2, word code3)
    {
    hancode code;

    code.n3.flag   = 1;
    code.n3.first  = code1;
    code.n3.second = code2;
    code.n3.third  = code3;
    return code.n2;
    }

/***************************************************************************/
static int char_class (int ch)
    {
    char *cp;
    static char s[] = "u7hk9yi;:nNjl0o['pm6De4358v/?bgfdtcr";
    static int  class[] = { C,C,C,C,C,C,C,C,C,C,C,C,C,C,C,C,
        C,C,C,V1,V1,V1,V1,V1,V1,V1,V2,V2,V3,V3,V4,V5,V5,V6,V6,V7 };

    static char bs[] ="F~#SAG$BW%^TX2aZ!xElswzWQq@R";
    static int  bclass[] = { B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,B1,
        B1,B2,B2,B2,B3,B4,B5,B5,B5,B6,B7,B8 };

    if ((cp = strchr(s, ch)) != NULL)
        return class[cp - s];
    else if ((cp = strchr(bs, ch)) == NULL)
        return ALPHA;
    return bclass[cp - bs];
    }
/***************************************************************************/
static int to_c5bit (int ch)
    {
    static char s[] = "u7hk9yi;:nNjl0o['pm";
    static int  c5bit_tbl[] = { C_GI, C_GI2, C_NI, C_DI, C_DI2, C_RI,
        C_MI, C_BI, C_BI2, C_SI, C_SI2, C_YI, C_JI, C_JI2, C_CHI,
        C_KI, C_TI, C_PI, C_HI };

    return c5bit_tbl[strchr(s,ch)-s];
    }

/***************************************************************************/
static int to_v5bit (int ch)
    {
    static char s[] = "fd6Dtce4/v3?b5g8r";
    static int  v5bit_tbl[] = { V_AH, V_AE, V_YA, V_YAE, V_UH, V_EH,
        V_YU, V_YE, V_OH,V_OH, V_YO, V_OO,V_OO, V_YOO, V_EU, V_EUI, V_YI };

    return v5bit_tbl[strchr(s,ch)-s];
    }

/***************************************************************************/
static int to_b5bit (int ch)
    {
    static char s[] = "xF~s#SAwG$BW%^Tz1EXq2a@Z!WQR";

    static int  b5bit_tbl[] = { B_GI, B_GI2, B_GS, B_NI, B_NJ,
        B_NH, B_DI, B_RI, B_RG, B_RM, B_RB, B_RS, B_RT, B_RP,
        B_RH, B_MI, B_BI, B_BI, B_BS, B_SI, B_SI2, B_YI, B_JI,
        B_CHI, B_KI, B_TI, B_PI, B_HI };

    return b5bit_tbl[strchr(s,ch)-s];
    }
```

```
/****************************************************************************/
static int rieul_complex (int ch)
    {
    char *sp;
    static char s[] = "xz1EqWQR";
    static int  rc[] = { B_RG, B_RM, B_RB, B_RB, B_RS, B_RT, B_RP, B_RH };

    if ((sp = strchr(s, ch)) == NULL)
        return NULL;
    return rc[sp - s];
    }
/****************************************************************************/
static  int conversionAscii(int ch)
    {
    char *sp;
    static char s[] = "MCVHJKLUIO&*(P{}])-=¦\\\"";
    static int  c[] = {0x0b30,0x2822,0x2822,0x2822,0x0231,0x0332,
                       0x0433,0x0534,0x0635,0x0637,0x0837,0x0938,
                       0x0a39,0x352f,0x0625,0x0221,0x353f,0x092a,
                       0x0a28,0x0b29,0x273b,0x273a,0x0c2D};
    if ((sp = strchr(s, ch)) != NULL)
        return c[sp - s];
    return ch;
    }
/****************************************************************************/
InsertTokenToQue(int x, int flag)
    {
    IORegs.x.DX = x;
    IORegs.h.AL = IORegs.h.DH;
    IORegs.h.AH = flag;
    add_que(IORegs.x.AX);
    IORegs.h.AL = IORegs.h.DL;
    IORegs.h.AH = flag;
    add_que(IORegs.x.AX);
    }
/****************************************************************************/
S13_X_CommonRtn(int class, int ch)
    {
    int store;

    InsertTokenToQue(code_stack[code_stack_ptr].code.n2, C_HANGUL);

    code_stack_ptr = 1;

    if(class == C)
        {
        last_c = to_c5bit(ch);
        store = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, V_FILL, B_FILL);
        status = code_stack[code_stack_ptr].station = S2;
        InsertTokenToQue(code_stack[code_stack_ptr].code.n2, U_HANGUL);
        }
    else if(class == V2 ¦¦ class == V3 ¦¦ class == V4)
        {
        switch(class)
            {
            case V2 : status =
                code_stack[code_stack_ptr].station = S1_1;  break;
            case V3 : status =
                code_stack[code_stack_ptr].station = S1_2;  break;
            case V4 : status =
                code_stack[code_stack_ptr].station = S1_3;  break;
            }

        last_v = to_v5bit(ch);
        store = code_stack[code_stack_ptr].code.n2
            = combine2(C_FILL, last_v, B_FILL);
        InsertTokenToQue(code_stack[code_stack_ptr].code.n2, U_HANGUL);
        }
    else
        {
        status = code_stack[code_stack_ptr].station = S1;

        if(class < B1)
            store = code_stack[code_stack_ptr].code.n2
                = combine2(C_FILL, to_v5bit(ch), B_FILL);
        else
            store = code_stack[code_stack_ptr].code.n2
                = combine2(C_FILL, V_FILL, to_b5bit(ch));

        InsertTokenToQue(code_stack[code_stack_ptr].code.n2, C_HANGUL);
        }
    return delete_que();
    }
/****************************************************************************/
DoubleVowel(int ch, int state)
    {
    switch(status)
        {
        case S1_1:
        case S2_1:
            switch (to_v5bit(ch))
                {
                case  V_AH : last_v = V_WA;   break;
                case  V_AE : last_v = V_WAE;  break;
                case  V_YI : last_v = V_WE;
                }
            break;
        case S1_2:
        case S2_2:
            switch (to_v5bit(ch))
                {
                case  V_UH : last_v = V_WO;   break;
                case  V_EH : last_v = V_WYE;  break;
                case  V_YI : last_v = V_WI;
                }
            break;
        case S1_3:
        case S2_3:
            last_v = V_EUI;
            break;
        }
    code_stack_ptr ++;
    status = code_stack[code_stack_ptr].station = state;
    }
/****************************************************************************/
Wait_InitOrBase(int class, int ch)
    {
    int     ret, tmp;

    if (class == C)                /* if ch == consonant */
```

```
        {
        last_c = 0;
        ret = InitialCharInput(ch);
        }
    else if (class >= B1 && class <= B8)
      {
      code_stack_ptr ++;  /* code_stack_ptr = 3,4 */
      tmp = U_HANGUL;
      switch (class)
          {
          case B2 : status =
              code_stack[code_stack_ptr].station = S3_1;  break;
          case B3 : status =
              code_stack[code_stack_ptr].station = S3_2;  break;
          case B4 : status =
              code_stack[code_stack_ptr].station = S3_3; break;
          default : status =
              code_stack[code_stack_ptr].station = S1;
              tmp = C_HANGUL;
          }
      last_b = ch;
      IORegs.x.DX = code_stack[code_stack_ptr].code.n2
          = combine2(last_c, last_v, to_b5bit(ch));
      InsertTokenToQue(IORegs.x.DX, tmp);
      ret = delete_que();
      }
  else
      {                         /* if ch == vowel */
      IORegs.x.DX = code_stack[code_stack_ptr].code.n2;
      InsertTokenToQue(IORegs.x.DX, C_HANGUL);
      tmp = U_HANGUL;
      status = S1;
      code_stack_ptr = 1;
      switch(class)
          {
          case V2 : status =
              code_stack[code_stack_ptr].station = S1_1;  break;
          case V3 : status =
              code_stack[code_stack_ptr].station = S1_2;  break;
          case V4 : status =
              code_stack[code_stack_ptr].station = S1_3;  break;
          default:
              tmp = C_HANGUL;
          }
      IORegs.x.DX = code_stack[code_stack_ptr].code.n2
          = combine2(C_FILL, to_v5bit(ch), B_FILL);
      InsertTokenToQue(IORegs.x.DX, tmp);
      ret = delete_que();
      }
    return(ret);
    }

/*****************************************************************************/
InitialCharInput(int ch)
    {
    static  char    DCC[] = "79:NO";
    static  char    s[] = "uk;nl";
    char    *sp;

    sp = strchr(s, ch);

    if(to_c5bit(ch) == last_c && (sp != NULL) && status == S2)
        ch = (char) DCC[sp - s];
    else
        {
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2;
        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        }

    code_stack_ptr = 1;
    status = code_stack[code_stack_ptr].station = S2;

    last_c = to_c5bit(ch);
    IORegs.x.DX = code_stack[code_stack_ptr].code.n2
        = combine2(last_c, V_FILL, B_FILL);
    InsertTokenToQue(IORegs.x.DX, U_HANGUL);
    return delete_que();
    }

/*****************************************************************************/
key_board_read()
    {
    union REGS inregs, outregs;

    inregs.h.ah = 0;
    int86(0x16, &inregs, &outregs);
    return(outregs.x.ax);
    }

/*****************************************************************************/
get_shift_status()
    {
    union REGS inregs, outregs;

    inregs.h.ah = 2;
    int86(0x16, &inregs, &outregs);
    return(outregs.h.al);
    }

/*****************************************************************************/
int MYGETCH()
    {
    #define SHIFT_MASK      0x03
    #define ALT_MASK        0x08

    char    status;
    union   {
            int     value;
            unsigned char    code[2];
            }  key;
    static  Toggle = SO;
    static  char NumericKey[] = "JKLUIO&*(M+\")";
    char    *sp="1234567890+-*",*cp;

    while (!kbhit() )
        status = get_shift_status();
    key.value = key_board_read();
    if(((status & 15)==8) && key.code[0] == 32 )
            return ( Toggle == SO) ? (Toggle = SI) : (Toggle = SO);
    else if(Toggle == SI) return key.value;
    else if((cp = strchr(sp,key.code[0])) &&
            ((key.code[1]>70 && key.code[1]<84) ||
            key.code[1] == 55))
            {
            key.code[0] =NumericKey[cp - sp];
            return(key.value);
            }
    else return (SHIFT_MASK & status) ?
                    (key.value&0xff00) | toupper(key.code[0])  :
                    (key.value&0xff00) | tolower(key.code[0]);
    }

/*****************************************************************************/
word get_hangul (void)
    {
    word temp, ch;
    int  class, ret;

    if ((temp=delete_que()) != NULL) return temp;
```

```
loop:  /*------------------------------ loop start here ----------------*/
ch = MYGETCH();

switch (ch)
    {
    case  SI :
        if (status <= S1)
            {
            status = S0;
            goto loop;
            }
        else
            {
            status = S0;
            IORegs.x.DX = code_stack[code_stack_ptr].code.n2;
            code_stack_ptr = 0;
            IORegs.h.AL = IORegs.h.DL;
            IORegs.h.AH = C_HANGUL;
            add_que(IORegs.x.AX);
            IORegs.h.AL = IORegs.h.DH;
            IORegs.h.AH = C_HANGUL;
            return IORegs.x.AX;
            }

    case  SO :
        if (status == S0)
            status = S1;
        goto loop;
    }

 if (status == S0) return ch;    /* if English Mode */

 if (ch == BS)
     if (status == S1)
         return ch;
     else
         {
         code_stack_ptr --;
         status = code_stack[code_stack_ptr].station;
         IORegs.x.DX = code_stack[code_stack_ptr].code.n2;
         InsertTokenToQue(IORegs.x.DX, U_HANGUL);
         return delete_que();
         }

 if ((class=char_class(ch)) == ALPHA)
     {
     ch = conversionAscii(ch);
     if (status==S1) return ch;
     else
      {
      IORegs.x.DX = code_stack[code_stack_ptr].code.n2;
      status = S1;
      code_stack_ptr = 0;
      InsertTokenToQue(IORegs.x.DX, C_HANGUL);
      add_que(ch);
      return delete_que();
      }
  }

switch (status)
  {
  case  S1 :
        code_stack_ptr = 1;            /* code_stack_ptr = 1 */
        if (class == C)                /* if ch == consonant */
            {
            last_c = to_c5bit(ch);
            IORegs.x.DX = code_stack[code_stack_ptr].code.n2
                = combine2(last_c, V_FILL, B_FILL);
            status = code_stack[code_stack_ptr].station = S2;
            InsertTokenToQue(IORegs.x.DX, U_HANGUL);
            ret = delete_que();
            }
        else if(class == V2 || class == V3 || class == V4)
            {
            switch (class)
                {
                case V2 : status =
                    code_stack[code_stack_ptr].station = S1_1;  break;
                case V3 : status =
                    code_stack[code_stack_ptr].station = S1_2;  break;
                case V4 : status =
                    code_stack[code_stack_ptr].station = S1_3;  break;
                }
            last_c = C_FILL;
            last_v = to_v5bit(ch);
            IORegs.x.DX = code_stack[code_stack_ptr].code.n2
                = combine2(last_c, last_v, B_FILL);

            InsertTokenToQue(IORegs.x.DX, U_HANGUL);
            ret = delete_que();
            }
        else
            {
            status = S1;
            if(class >= B1 && class <= B8)
                IORegs.x.DX = combine2(C_FILL, V_FILL, to_b5bit(ch));
            else IORegs.x.DX = combine2(C_FILL, to_v5bit(ch), B_FILL);
            InsertTokenToQue(IORegs.x.DX, C_HANGUL);
            ret = delete_que();
            }
        break;

    case  S2 :
        if (class == C)                  /* if ch == consonant */
            ret = InitialCharInput(ch);
        else if(class >= B1 && class <= B8)
            {
            status = code_stack[code_stack_ptr].station = S1;
            IORegs.x.DX = code_stack[code_stack_ptr].code.n2;
            InsertTokenToQue(IORegs.x.DX, C_HANGUL);
            IORegs.x.DX = combine2(C_FILL, V_FILL, to_b5bit(ch));
            InsertTokenToQue(IORegs.x.DX, C_HANGUL);
            ret = delete_que();
            }
        else                             /* if ch == vowel */

         code_stack_ptr ++;  /* code_stack_ptr = 2 */
         switch (class)
             {
             case V2 : status =
                 code_stack[code_stack_ptr].station = S2_1;  break;
             case V3 : status =
                 code_stack[code_stack_ptr].station = S2_2;  break;
             case V4 : status =
                 code_stack[code_stack_ptr].station = S2_3;  break;
```

```
            default : status =
                code_stack[code_stack_ptr].station = S3;
            }
        last_v = to_v5bit(ch);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, B_FILL);

        InsertTokenToQue(IORegs.x.DX, U_HANGUL);
        ret = delete_que();
        }
    break;

case S3 :
    ret = Wait_InitOrBase(class, ch);
    break;

case S1_1:
    if (class !=V5 && class!=V7)
        ret = S13_X_CommonRtn(class, ch);
    else
        {
        DoubleVowel(ch, S1);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(C_FILL, last_v, B_FILL);
        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        ret = delete_que();
        }
    break;

case S1_2:
    if (class!=V6 && class!=V7)
        ret = S13_X_CommonRtn(class, ch);
    else
        {
        DoubleVowel(ch, S1);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(C_FILL, last_v, B_FILL);
        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        ret = delete_que();
        }
    break;

case S1_3:
    if (class != V7)
        ret = S13_X_CommonRtn(class, ch);
    else
        {
        DoubleVowel(ch, S1);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(C_FILL, last_v, B_FILL);

        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        ret = delete_que();
        }
    break;

case S2_1 :
    if (class !=V5 && class!=V7)
        ret = Wait_InitOrBase(class, ch);
    else
        {
        DoubleVowel(ch, S3);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, B_FILL);

        InsertTokenToQue(IORegs.x.DX, U_HANGUL);
        ret = delete_que();
        }
    break;
```

```
case S2_2:
    if (class!=V6 && class!=V7)
        ret = Wait_InitOrBase(class, ch);
    else
        {
        DoubleVowel(ch, S3);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, B_FILL);

        InsertTokenToQue(IORegs.x.DX, U_HANGUL);
        ret = delete_que();
        }
    break;

case S2_3:
    if (class != V7)
        ret = Wait_InitOrBase(class, ch);
    else
        {
        DoubleVowel(ch, S3);
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, B_FILL);

        InsertTokenToQue(IORegs.x.DX, U_HANGUL);
        ret = delete_que();
        }
    break;

case S3_1 :
    if (class != B6)
        ret = S13_X_CommonRtn(class, ch);
    else
        {
        temp = (to_b5bit(last_b) == B_GI) ? B_GS : B_BS;
        code_stack_ptr ++;  /* code_stack_ptr = 3,4 */
        status = code_stack[code_stack_ptr].station = S1;
        last_b = ch;
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, temp);
        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        ret = delete_que();
        }
    break;

case S3_2 :
    if (class!=B7 && class!=B8)
        ret = S13_X_CommonRtn(class, ch);
    else
        {
        temp = (to_b5bit(ch) == B_JI) ? B_NJ : B_NH;
        code_stack_ptr ++;  /* code_stack_ptr = 3,4 */
        status = code_stack[code_stack_ptr].station = S1;

        last_b = ch;
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, temp);
        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        ret = delete_que();
        }
    break;

case S3_3 :
    if ((temp=rieul_complex(ch)) == NULL)
        ret = S13_X_CommonRtn(class, ch);
    else
        {
        code_stack_ptr ++;  /* code_stack_ptr = 3,4 */
        status = code_stack[code_stack_ptr].station = S1;
```

```
        last_b = ch;
        IORegs.x.DX = code_stack[code_stack_ptr].code.n2
            = combine2(last_c, last_v, temp);
        InsertTokenToQue(IORegs.x.DX, C_HANGUL);
        ret = delete_que();
                }
            break;
        }
    return(ret);
    }
```

(a) hiokbw.c

```
/**************************************************************************\
        Module name : test3kbd.c
\**************************************************************************/

#include "hiokbw.c"
#define ESC 0x011b

/**************************************************************************/
void main()
        {
        word ch;

        while ((ch=get_hangul()) != ESC)
                {
                if ((ch>>8) == 0xF0)
                        {
                        putchar(ch);
                        putchar(get_hangul());
                        putchar(BS);
                        putchar(BS);
                        }
                else if ((ch & 0x00FF) == 0x0d)
                        putchar(0x0a);
                else putchar(ch);
                }
        }
```

(b) test3kbd.c

□ 2벌식 자판배열에 따른 한글입력 자소구분

① C1={ㄸ, ㅃ, ㅉ} : 초성에만 올 수 있는 쌍자음
② C2={ㄲ, ㄷ, ㅆ, ㅇ, ㅊ, ㅋ} : 초성과 종성에 올 수 있는 자음
③ C3={ㄱ, ㅂ} : 초성과 종성에 올 수 있으며 다음에 오는 "ㅅ"과 복합자음의 구성이 가능한 자음
④ C4={ㄴ} : 초성과 종성에 올 수 있으며 다음에 오는 "ㅈ, ㅎ"과 복합자음의 구성이 가능한 자음
⑤ C5={ㄹ} : 초성과 종성에 올 수 있으며 다음에 오는 "ㄱ, ㅂ, ㅁ, ㅌ, ㅍ, ㅅ, ㅎ"과 복합자음의 구성이 가능한 자음
⑥ C6={ㅁ, ㅌ, ㅍ} : 초성과 종성에 올 수 있으며 "ㄹ"과 결합하여 복합자음의 구성이 가능한 자음
⑦ C7={ㅅ} : 초성과 종성에 올 수 있으며 "ㄱ, ㄹ"과 결합하여 복합자음의 구성이 가능한 자음
⑧ C8={ㅈ} : 초성과 종성에 올 수 있으며 "ㄴ"과 결합하여 복합자음의 구성이 가능한 자음
⑨ C9={ㅎ} : 초성과 종성에 올 수 있으며 "ㄴ, ㄹ"과 결합하여 복합자음의 구성이 가능한 자음
⑩ V1={ㅑ, ㅒ, ㅕ, ㅖ, ㅛ, ㅠ} : 복합모음 구성이 불가능한 모음
⑪ V2={ㅗ} : 다음에 오는 "ㅏ, ㅐ, ㅣ"와 결합하여 복합모음의 구성이 가능한 모음
⑫ V3={ㅜ} : 다음에 오는 "ㅓ, ㅔ, ㅣ"와 결합하여 복합모음의 구성이 가능한 모음
⑬ V4={ㅡ} : 다음에 오는 "ㅣ"와 결합하여 복합모음의 구성이 가능한 모음
⑭ V5={ㅏ, ㅐ} : 모음 "ㅗ"와 결합하여 복합모음의 구성이 가능한 모음
⑮ V6={ㅓ, ㅔ} : 모음 "ㅜ"와 결합하여 복합모음의 구성이 가능한 모음
⑯ V7={ㅣ} : 모음 "ㅡ"와 결합하여 복합모음의 구성이 가능한 모음
⑰ SW : 한/영 입력 상태를 전환하는 키
⑱ α : 자판에 있는 숫자, 특수문자, Function키, 콘트롤 키, Alt키와의 조합 등 자판에서 영어 알파벳 키를 제외한 모든 키

□ 3벌식 자판배열에 따른 한글입력자소구분

① C={모든 초성} : 단자음, 쌍자음으로 초성에 올 수 있는 모든 자음
② V1={ㅑ, ㅒ, ㅕ, ㅖ, ㅛ, ㅠ} : 복합모음 구성이 불가능한 모음
③ V2={ㅗ} : 다음에 오는 "ㅏ, ㅐ, ㅣ"와 결합하여 복합모음의 구성이 가능한 모음
④ V3={ㅜ} : 다음에 오는 "ㅓ, ㅔ, ㅣ"와 결합하여 복합모음의 구성이 가능한 모음
⑤ V4={ㅡ} : 다음에 오는 "ㅣ"와 결합하여 복합모음의 구성이 가능한 모음
⑥ V5={ㅏ, ㅐ} : 모음 "ㅗ"와 결합하여 복합모음의 구성이 가능한 모음
⑦ V6={ㅓ, ㅔ} : 모음 "ㅜ"와 결합하여 복합모음의 구성이 가능한 모음
⑧ V7={ㅣ} : 모음 "ㅡ"와 결합하여 복합모음의 구성이 가능한 모음
⑨ B1={ㄲ, ㄳ, ㄵ, ㄶ, ㄷ, ㄺ, ㄻ, ㄼ, ㄽ, ㄾ, ㄿ, ㅀ, ㅄ, ㅆ, ㅇ, ㅊ, ㅋ} : 다음에 입력되는 자소와 결합하지 않는 받침
⑩ B2={ㄱ, ㅂ} : 다음에 오는 받침 "ㅅ"과 결합하여 복합받침의 구성이 가능하거나 받침 "ㄹ"과 결합하여 복합받침의 구성이 가능한 받침
⑪ B3={ㄴ} : 다음에 오는 받침 "ㅈ, ㅎ"과 결합하여 복합받침의 구성이 가능한 받침
⑫ B4={ㄹ} : 다음에 오는 받침 "ㄱ, ㅁ, ㅂ, ㅅ, ㅌ, ㅍ, ㅎ"과 결합하여 복합받침의 구성이 가능한 받침
⑬ B5={ㅁ, ㅌ, ㅍ} : 받침 "ㄹ"과 결합하여 복합받침의 구성이 가능한 받침
⑭ B6={ㅅ} : 받침 "ㄱ, ㄹ, ㅂ"과 결합하여 복합받침의 구성이 가능한 받침
⑮ B7={ㅈ} : 받침 "ㄴ"과 결합하여 복합받침의 구성이 가능한 받침
⑯ B8={ㅎ} : 받침 "ㄴ, ㄹ"과 결합하여 복합받침의 구성이 가능한 받침
⑰ SW : 한/영 입력상태를 전환하는 키
⑱ α : 자판에 있는 숫자, 특수문자, Function키, 콘트롤 키, Alt키와의 조합 등 자판에서 영어 알파벳 키를 제외한 모든 키

●유틸리티 소개●

효율적으로 메모리를 관리해주는 유틸리티

소프트웨어 캐러젤

정승민 / 아주대 수학과 HAL

소프트웨어 캐러젤은 간단한 키조작으로 최대 10개까지의 소프트웨어 사이를 옮겨다니면서 원하는 프로그램을 실행시킬 수 있어 빠르고 신속하게 업무를 처리할 수 있으며, 640KB의 기본 메모리 외에도 확장메모리도 사용할 수 있어 효율적으로 메모리를 관리할 수 있다.

16비트 PC OS의 대부분을 차지하는 PC DOS(MS DOS와 구별되는 이름이지만 기능이 거의 유사하므로 앞으로 PC DOS라 하면 둘다를 지칭하는 말로 하겠다)에는 커다란 약점이 있다. 즉 메모리의 사용가능한 최대량이 640KB이기 때문에 여러개의 소프트웨어를 실행시킬 수가 없다는 점과 램상주 소프트웨어의 표준화가 이루어지지 않고 있기 때문에 램상주 프로그램간의 키충돌이나 시스템 다운이 자주 일어난다는 점 등이다.

그러나 하늘이 무너져도 솟아날 구멍이 있듯이 윈도우, 데스크뷰, 멀티링크 등의 소프트웨어들이 등장하면서 PC DOS가 가지고 있는 약점들을 해결하게 되었다. 이들 프로그램은 나름대로의 장점을 가지고 있지만 모두를 한꺼번에 전부 소개하기에는 지면의 부담이 따르므로 이번 달에는 PC DOS의 약점을 덮어주는 유틸리티인 소프트로직(SoftLogic)사의 소프트웨어 캐러젤(Software Carousel)을 소개한다.

특징

소프트웨어 캐러젤은 인스톨시키는 프로그램인 INSTALL.BAT, 캐러젤 본래의 화일인 CAROUSEL.PGM, 캐러젤의 여러 정보를 저장하고 있는 CAROUSEL.OPT, 캐러젤을 제거해주는 프로그램인 UNINSTAL.BAT와 유저 ID를 기록하는 ID.CAR 등의 6개 화일로 구성되어 있으며, 앞에서 언급된 여러 문제를 깨끗이 해결해주는 기특한(?) 유틸리티이다. 유틸리티보다는 OS 소프트웨어쪽에 더 가깝다고도 할 수 있는데, 소프트웨어 캐러젤의 장점은 간단한 키조작으로 최대 10개까지의 소프트웨어 사이를 옮겨다니면서 실행시킬 수 있다는 것이다.

예를 들면 사무실에서 가장 많이 사용하는 소프트웨어로 워드프로세서와 스프레드시트가 있다고 하자. 만약 워드프로세서를 사용하다가 스프레드시트를 실행시키고자 한다면 대개 입력중이었던 것을 저장하고 워드프로세서를 종료시킨 다음 스프레드시트를 실행시켜야 하는 등의 번거로운 작업을 해야 한다.

이럴 때 소프트웨어 캐러젤을 사용하면 두 프로그램 사이를 이동하면서 빠르고 편안하게 원하는 업무를 처리할 수 있다.

또한 소프트웨어 캐러젤의 장점은 소프트웨어간의 이동외에도 메모리의 효과적인 사용에 있다.

640KB의 기본 메모리를 사용하는 것이외에 증설(Extended) 메모리나 확장 (Expanded) 메모리, 하드디스크나 플로피디스크 등도 사용할 수 있다.

인스톨과 실행

소프트웨어 캐러젤의 인스톨 절차는 매우 간단하다. 단지 소프트웨어 캐러젤 디스켓을 드라이브에 넣고 다음과 같은 명령을 내리기만 하면 된다.

```
INSTALL A N
```

여기서 N은 대상드라이브 이름이고 주의할 점은 드라이브 이름 뒤에 콜론(:)을 쓰지 않는다는 사실이다. 인스톨 명령을 내리면 소프트웨어 캐러젤은 화면 1과 같이 소유주의 이름을 묻는다. 그리고 나서 실행화일을 대상 디스크에 복사하게 된다.

인스톨이 끝났으면 이제 소프트웨어 캐러젤을 실행시켜 보자.

우선 소프트웨어 캐러젤을 인스톨한 디스크로 가서

```
CAROUSEL
```

라고 하면 소프트웨어 캐러젤이 실행된다. 처음 실행될 때 소프트웨어 캐러젤은 사용할 소프트웨어의 메모리를 얼마로 잡을 것인가에 대한 질문을 해온다. 이 질문은 메모리 파티션을 나눌 때 허용가능한 메모리 크기와 관련 있으므로 메모리 크기를 변경하고 싶을 때는 질문에 Y로 답한다. 메모리에 관련된 질문을 마치고 나면 다시 도스 상태로 되돌아간다.

이후부터는 소프트웨어 캐러젤이 작동되는 상태이다. 이제부터는 각 메모리의 파티션사이를 이동할 수 있다. 메모리 파티션이란 소프트웨어 캐러젤이 메모리(또는 가상 메모리)를 분할했을 때 각각의 메모리영역을 말한다. 파티션은 앞에서도 언급했지만 최대 10개까지 가능하다.

도스 프롬프트의 바로 윗줄을 보면 파티션에 관한 정보가 있다. 화면 2를 보면 "Memory Partition 1~256K"라는 메시지가 나타난다. 여기서 "Memory Partition 1"은 현재의 메모리 파티션 설명이고(이것은 나중에 메뉴에서 수정할 수 있다) "256K"라는 것은 현재의 메모리 파티션 크기이다.

여기서 시스템을 부팅시키기 위해 〈Ctrl〉, 〈Alt〉, 〈Del〉 세 키를 동시에 눌러보자. 평상시와 같이 바로 부팅되지 않고 소프트웨어 캐러젤이 부팅여부에 대한 질문을 하게 된다. 이것은 부주의로 인한 부팅을 막기 위한 방지책이다.

이제는 파티션간의 이동을 해보자. 파티션간의 이동은 〈Alt〉키와 해당 파티션의 펑션키를 동시에 누름으로써 할 수 있다(이 때 〈Alt〉키 이외에 〈Ctrl〉키, 양쪽 〈Shift〉키, 브레이크키 등을 사용할 수 있는데 디폴트는 〈Alt〉키이다. 물론 이 핫키(Hotkey)는 뒤의 옵션화일에서 다른 것으로 지정할 수도 있다).

파티션은 펑션키의 번호로 지정할 수 있는데 파티션의 지정은 뒤에서 설명하겠다.

현재의 파티션은 1번인데, 예를 들어서 3번 메모리 파티션으로 이동한다고 했을 때, 〈Alt〉키와 〈F3〉키를 동시에 누르면 되는 것이다.

〔화면 1〕 인스톨 후 소유주를 묻는 화면

```
        This Software Carousel software is copyright 1986
              by SoftLogic Solutions, Inc.
                    Manchester, NH

This software is provided without copy protection for your convenience,
but the software is explicitly licensed for use on ONLY ONE PC.

You, or your company, are the licensee of the Software Carousel software.
Please enter your name or your company's name as the licensee.


Enter name of licensee: DOOLY
Name entered O.K. ? (Y or N) y
```

〔화면 2〕 실행후 도스상태 화면

```
Current date is Tue  1-01-1980
Enter new date (mm-dd-yy):
Current time is  0:24:59.41
Enter new time:


The IBM Personal Computer DOS
Version 3.30 (C)Copyright International Business Machines Corp 1981, 1987
             (C)Copyright Microsoft Corp 1981, 1986

Memory Partition 1 - 256K
A:\>
```

메모리 관리

소프트웨어 캐러젤이 사용할 수 있는 가상 메모리 장치는 크게 다음의 4가지가 있다.

(1) 로우워 램(Lower Ram)

도스가 사용가능한 최대 메모리인 640KB를 말한다. 여기서 도스가 앞의 얼마간을 사용하게 되므로 실제로 소프트웨어 캐러젤이 사용가능한 메모리는 600KB가 조금 안된다. 여기서도 그렇고 다음의 3가지 메모리형태에서도 모두 적용되는 사실인데, 메모리 지정시의 단위는 1KB가 아니라 16KB씩이라는 것을 명심해 주기 바란다. 예를 들어서 120K를 메모리 지정에서 지정했을 때, 소프트웨어 캐러젤은 자동적으로 128K로 변환(?)해서 입력을 받는다.

(2) 증설 램(Extended Ram)

AT이거나 AT 호환기종이면 증설램이라는 메모리가 있는데 AT 시스템의 메모리 크기에서 640KB를 뺀 수치가 보통 증설 메모리의 크기가 된다. 예를 들어서 1MB의 메모리를 가진 AT라면 384KB이다. 그리고 증설 메모리는 일반적으로 VDISK 등을 사용하여 램디스크로 사용하는 경우가 대부분인데, 독자들이 형평(?)을 고려하여 소프트웨어 캐러젤이냐 램디스크냐를 선택하기 바란다. 이것은 최대 16MB까지 사용가능하다.

(3) 확장 램(Expanded Ram)

증설 램과 발음과 철자가 비슷하여 항상 IBM PC 유저의 머리를 어지럽게 하는데, 이것은 메모리를 증설하는 방법의 일종으로 로터스, 인텔, 마이크로소프트가 공동으로 제정한 특정한 방법에 의하여 메모리를 확장하는 방법을 말한다. 소프트웨어 캐러젤의 메모리를 사용하면 오토캐드, 워드스타, 로터스 등의 큰 프로그램을 연이어 실행시키면서도 각 메모리 파티션간의

이동에 별로 시간이 걸리지 않는다(디스크화일을 이용한 파티션간 이동은 하드 디스크를 사용해도 시간이 꽤 걸린다). 이것은 최대 8MB까지 가능하다.

(4) 디스크 화일

이것은 플로피나 하드디스크의 여백을 가상 램으로 사용할 수 있게 해주는 것이다. 최대 6.4MB까지 가능하다.

메뉴 사용법

소프트웨어 캐러젤은 실행중 각종 수치를 조절할 수 있다. 대표적인 예로서 각 메모리 파티션의 크기를 사용가능한 메모리범위내에서 바꿀 수가 있다. 또 각 메모리 파티션에다 고유의 이름을 붙일 수도 있다.

우선 각종 소프트웨어 캐러젤의 메뉴 옵션을 사용하기 위해서는 메뉴를 호출해야만 한다. 소프트웨어 캐러젤의 메뉴를 호출하려면 〈Alt〉키와 스페이스키를 동시에 누르면 소프트웨어 캐러젤의 메뉴가 화면 3과 같이 나오게 된다.

〔화면 3〕 메뉴화면

```
<> S o f t w a r e   C a r o u s e l  ver 2.00<>      Copyright 1986 by
                                                     SoftLogic Solutions, Inc.
                                                        Manchester, NH
 ═══════════O P T I O N S═══════════
  E    Exit from this menu                          ═H E L P  Information═
  C    Change Memory Information
  R    Rename a Memory Partition
  K    Kill a Program                                 This option will leave
  H    Help Display On & Off                         this menu and return to
  Fx   F1 - F10 Switches to the                      the program you came here
          Program as in SETUP Box                    from, in partition F1.
  ↑↓ Move the hi-lite. ↵ Selects

                                  ═Your  S E T U P  Now══════════Total  496K═
                                   F1.   Memory Partition 1,  Un-named     256K
                                   F2.   Memory Partition 2,  Un-named      64K
                                   F3.   Memory Partition 3,  Un-named      64K
                                   F4.   Memory Partition 4,  Un-named      64K
                                   F5.   Memory Partition 5,  Un-named      00K
                                   F6.   Memory Partition 6,  Un-named      00K
                                   F7.   Memory Partition 7,  Un-named      00K
                                   F8.   Memory Partition 8,  Un-named      00K
                                   F9.   Memory Partition 9,  Un-named      00K
                                   F10.  Memory Partition 10, Un-named      00K
                                  ══════════════════════════════Free   048K═
```

먼저 메뉴의 생김새(?)를 한번 살펴보자. 화면 왼쪽위에는 메뉴에서 사용할 수 있는 옵션들이 나열되어 있다. 각 옵션의 기능을 알아보자.

① E: 메뉴를 빠져나간다.
② C: 메모리 파티션의 크기를 바꾼다.
③ R: 메모리 파티션의 이름을 바꾼다.
④ K: 현재의 파티션에 있는 프로그램을 메모리에서 없앤다.
⑤ H: 메뉴 화면 오른쪽위의 헬프의 표시여부를 온/오프시킨다.
⑥ 〈F1〉-〈F10〉: 해당 메모리 파티션으로 이동한다.

여기서 C는 사용할 소프트웨어의 메모리 파티션 크기를 지정하거나 변경할 때 사용하는 것이다. 우선 커서를 화살표키로 움직여서 C에 위치시킨 후에 리턴키를 치게 되면 수치를 입력받을 준비가 끝난다. 이때 숫자키는 메모리 파티션의 크기를 입력하게 되고 화살표키로는 메뉴화면 오른쪽 아래부분에서 해당 메모리 파티션을 지정하게 된다.

우선 바꾸고자 하는 메모리 파티션으로 커서를 이동시킨 후에 원하는 파티션의 크기를 입력한 후 리턴키를 누르면 해당 메모리 파티션의 크기가 바뀌어진다. 이 때 오른쪽 아래부분의 각 메모리 파티션 정보가 나온 박스가 있는데 여기의 제일 하단 부분에 현재 소프트웨어 캐러젤이 더 사용할 수 있는 남은 메모리의 크기가 나타난다. 이 범위내에서 파티션의 크기를 입력해야 한다.

이렇게 해서 램의 크기에 제약받지 않고 각 파티션 별로(물론 캐러젤에서 지정한 한도내에서) 원하는 소프트웨어를 실행시킬 수 있다.

R은 각 파티션의 이름을 고치고자 할 때 사용하게 되는데, 이것의 사용법도 ②의 사용법과 거의 같다. 다른 점은 파티션의 크기를 입력하는 대신에 파티션의 이름을 입력하는 점뿐이다.

이제 원하는 내용을 다 고쳤으면 'E' 옵션을 사용하여 메뉴를 빠져나간다. 이때 소프트웨어 캐러젤이 현재 바꾼 내용을 옵션화일에 저장할 것인가에 관해서 물어본다. 나중에 자신이 지금 바꾼 파티션 상태로 다시 사용하고 싶다면 이 파티션 정보를 저장해 두어야 한다.

CAROUSEL.OPT

이 화일은 화면 4와 같이 소프트웨어 캐러젤이 필요로 하는 각종 정보를 저장하고 있는 화일이다. 즉, 각 메모리 파티션의 크기, 이름 등을 비롯하여 핫키 등의 스캔코드가 입력되어 있다. 여기에서 필요한 정보는 사이드킥의 노트패드나 PC-Write, 워드스타의 넌도큐멘트를 이용하여 수정하기 바란다.

우선 소프트웨어 캐러젤이 사용하는 장치에 관하여 알아보자. 여기서 XXXX는 임의의 자연수이다.

① LOWER RAM＝XXXX K

640KB의 도스 메모리 중에서 남겨

〔화면 4〕 CAROUSEL. OPT의 화면

```
FILE = 0
HOT KEY = CTRL
MENU KEY = 1
TOGGLE KEY = 2

SIZE 1  = 256
SIZE 2  = 240
NAME 1  = APPLICATION
NAME 2  = MINI RESIDENT
```

두는 메모리의 크기를 여기에 적는다.

② EXPANDED RAM = XXXX K

확장 EMS 보드에서 얼마만큼의 메모리를 소프트웨어 캐러젤이 사용할 것인가에 대하여 여기에 적어둔다. 만약 CAROUSEL.OPT에 이 정보가 기록되어 있지 않으면서 EMS 보드를 소유하고 있다면 소프트웨어 캐러젤은 이것에 관하여 실행시킬 때 사용여부를 물어본다.

③ EXTENDED RAM = XXXX K

AT의 증설 메모리를 얼마만큼 사용할 것인가에 관한 정보를 여기에 기록한다.

④ DRIVE = N
FILE SIZE = XXXX K

디스크의 화일을 지정해 주는 부분이다. 여기서 N이 사용할 드라이브 이름이고, "FILL SIZE="가 화일의 크기이다.

⑤ NAME X = 파티션 이름

각 파티션의 이름을 지정하는 부분이다. 여기서 X는 파티션의 번호이다.

⑥ SIZE X = XXXX K

각 파티션의 크기를 지정하는 부분이다.

여기서도 X는 파티션의 번호이다. 다음으로 소프트웨어 캐러젤이 사용하는 핫키에 관한 정보에 대하여 알아보자.

⑦ HOT KEY = { ALT¦CTRL¦LSHIFT¦RSHIFT¦BREAK }

소프트웨어 캐러젤의 명령을 수행하기 위한 핫키의 정의부분이다.

위의 5개의 키이름중에서 하나를 쓰면 된다. 디폴트는 〈Alt〉키이다.

⑧ MENU KEY = SS

핫키와 같이 사용되어 소프트웨어 캐러젤의 메뉴를 부르게 되는 키이다. 여기서 SS는 메뉴키에 사용되어질 키의 스캔코드값이다.

스캔코드값은 각 컴퓨터 기종의 테크니컬 레퍼런스 매뉴얼을 참고하기 바란다. 프로그램 수행 도중에 이 키를 누르면 소프트웨어 캐러젤의 메뉴가 나온다.

⑨ TOGGLE KEY = SS

이것은 ⑦의 핫키와 같이 사용되어 바로 다음의 메모리 파티션으로 전환시켜 주는 기능을 말한다. 여기서도 SS는 키보드의 스캔코드 값이다.

⑩ JOB X KEY = SS

이것은 현재의 작업을 가로채서 바로 파티션 번호 X로 이동시키는 키이다. 여기서 X는 파티션 번호이고 SS는 키보드 스캔코드 값이다. 역시 ⑦의 핫키와 같이 눌러 지정된다.

⑪ PROGRAM X = 실행시킬 프로그램 이름

여기서 X는 파티션의 번호이며 소프트웨어 캐러젤이 실행과 동시에 각 파티션에 자동적으로 프로그램을 실행시킬 수 있게 하는 명령이다.

그래픽과 소프트웨어 캐러젤

소프트웨어 캐러젤은 그래픽을 지원하지는 못한다. 여기에서 가장 문제가 되는 것중의 하나가 바로 그 유명한 "보석글"이다.(특히 NKP 한글을 이용할 경우) 그래서 필자는 편법을 사용하고 있는데 그래픽 프로그램을 사용하다가 파티션간에 이동을 했을때, 그래픽이 깨지거나 그래픽 화면을 볼 수 없는 것을 막기 위하여 "HERC.COM"이라는 조그만 유틸리티를 이용하고 있다. 이것은 허큘리스 그래픽 카드에서 지원되는 거의 모든 모드간의 이동을 자유롭게 하는 것으로서 상당히 훌륭한 유틸리티이다. 필자는 또한 IBM LOGO를 사용할 때에도 이 "HERC.COM"의 덕을 톡톡히 보았고 보석글로 이번 호를 쓰면서도 이 조그만 유틸리티 덕을 보고 있다.

NKP 한글같은 그래픽 프로그램을 소프트웨어 캐러젤과 함께 사용할 때에는 먼저 "HERC.COM"을 실행시키고 나서 NKP를 실행시킨다. 그 후에 한글 화면이 나오지 않을 경우에는 〈Alt〉키와 〈5〉키를 누르면 된다. 참고로 〈Alt〉키와 〈0〉키를 누르면 "HERC.COM"의 헬프가 나온다.

좋은 소프트웨어는 많은 사람들이 여러번 사용해 보아야 그 진가를 알 수 있으며 기능을 향상시킬 수 있다는 것이 필자의 경험에서 얻은 결론이었다. 우리나라에서 직접 만든 소프트웨어가 아직은 드문 상황일때 외국의 좋은 소프트웨어를 값싸게(?) 실행시켜 보는 것도 나쁜 일만은 아닐 것이다.

CP/M-86과 MS-DOS를 인터페이스 시켜주는

화일전송 툴

김종석 / 챌린즈 기술부

지금까지 MS-DOS와 CP/M-86간은 서로간의 트랙 및 섹터구조 등이 달라 데이터의 호환성이 전혀 없었다. 따라서 DOS상에서 CP/M 화일을 읽어들이거나 CP/M상에서 DOS화일을 읽어들이는 것이 불가능하였다. 그러나 다음에 소개하는 프로그램은 이러한 불편을 말끔히 씻어줄 뿐 아니라 전송된 화일의 목록이나 텍스트 화일 등도 확인할 수 있게 해 준다.

들어가는 말

이 프로그램은 지난호의 특집 CP/M-86의 분석과 연관된 프로그램으로 지난호를 보신 분들은 이 프로그램이 왜 필요한지 쉽게 이해할 수 있으리라 생각한다. 이것은 바로 CP/M-86 과 MS-DOS 간의 인터페이스를 위한 것으로 화일 전송에 관한 프로그램을 소개하고자 하는 것이다.

먼저 여기에서 소개하는 프로그램은 모두 MS-DOS 상에서 Turbo Pascal 5.0으로 작성되었음과 DOS 상에서 실행됨을 미리 밝혀둔다. 또한 이 프로그램들에 대한 설명은 지난호의 내용을 기준하여 전개될 것이므로 중복되는 내용에 대한 설명은 가급적 피하고자 했다. 이번호만 구입하여 보는 독자들께는 양해를 구하며 가능하면 CP/M-86에 대한 참고 문헌을 참조해 주기 바란다.

필요한 화일

다음에 소개하는 프로그램은 모두 세 부분으로 되어 있는데, 첫번째가 DOS 시스템 상에서 CP/M-86 시스템 디스크에 있는 화일을 DOS 화일로 또는 DOS 화일을 CP/M 화일로 전송하는 프로그램이고, 두번째가 CP/M 상의 화일목록을 표시하는 프로그램, 세번째가 이 화일목록중 한 화일의 리스트를 표시하는 프로그램이다.

다시 말하면 **리스트 1**의 TRANSFER. PAS 프로그램은 지난호의 마지막 부분(151페이지)에서 언급한 MS-DOS 와 CP/M-86 상호간의 화일 전송 프로그램으로서 한 시스템에서 상이한 운영체제 디스켓 사용으로 인한 단절을 해결해 주는 유틸리티 프로그램이다.

또한 이렇게 전환된 화일목록은 다시 새로운 운영체제(OS)를 통해서 확인해야 하는 불편함이 발생하는데 현 운영체제에서도 다른 운영체제로 전송된 화일의 목록과 리스트를 표시할 수 있는 것이 **리스트 2**의 LS. PAS 프로그램(DOS 상에서 CP/M-86의 화일목록을 출력하는 프로그램)과 **리스트 3**의 CAT. PAS 프로그램(DOS 상에서 CP/M-86의 텍스트 화일을 화면으로 출력하는 프로그램)이다.

사용법

(1) TRANSFER

이 프로그램은 반드시 두대 이상의 디스크 드라이브(1 FDD, 1 HDD 또는 2 FDD)를 필요로 하며, MS-DOS 혹은 PC-DOS 시스템을 부팅시킨후 다음과 같은 순서 및 형식으로 명령어 라인을 입력한다.

TRASNSFER [/S] 소스화일명 [목적화일명]

즉, 옮기고자 하는 화일의 이름을 입력하고 다음에 옮겨져 기록될 화일의 이름을 입력하면 TRANSFER 화일에 따라 전송이 이루어진다. 이것은 DOS의 COPY 명령어와 유사한 형식이다. 만일 목적화일명을 입력하지 않으면, 전송될 화일명(소스화일명)이 디폴트로 목적화일명이 된다. 그리고드라이브에 대한 지정이 없을 경우, A : 드라이브가 디폴트 전송 드라이브가 되고 현재 사용중인 드라이브가 전송받을 목적 드라이브가 된다.

선택 사항(/S)으로는 다음과 같은 스위치가 있다.

- **/D** : CP/M-86 화일을 DOS 화일로 전송
- **/C** : DOS 화일을 CP/M-86 화일로 전송
- **/B** : 이진(Binary) 화일의 전송
- **/T** : 텍스트(Text) 화일의 전송

예를 들어 C: 드라이브상태에서 A: 드라이브에 있는 DUMPDISK. A86 의 CP/M 화일을 DOS 상태에 있는 B:드라이브의 디렉토리 \USR 밑에 DUMP. A86의 화일명으로 전송시키고자 한다면 다음과 같이 한다(화면 1 참조).

```
C:\SYJ>TRANSFER /D /B A: B:\USR\
```

(2) LS

이 화일은 DOS상에서 DIR 명령을 대신해 주는 프로그램으로 CP/M시스템이 들어있는 디스크의 화일목록을 보고자 할 때 사용하고, 특히 화일 전송시 CP/M디스크에 전송된 화일목록을 확인하고자 할 때 사용하면 편리하다. 이 프로그램 또한 두대 이상의 디스크 드라이브를 필요로 한다.

그리고 이 화일의 사용형식은 다음과 같다(화면 2 참조).

LS

(이 프로그램은 CP/M 시스템 디스크가 A: 드라이브에 있는 것으로 가정하였다.)

〔화면 1〕 CP/M화일을 DOS화일로 전송

```
C:\SYJ>TRANSFER /D /B A:DUMPDISK.A86 B:\USR\DUM.A86

Transfer CP/M file to DOS file : DUMPDISK.A86 -> B:\USR\DUMP.A86

Press any key!
CP/M file transfered.

C:\SYJ>
```

〔화면 2〕 LS화일의 실행예

```
C:\SYJ>LS

Drive A:

CPM      SYS    97 Recs      13 Kbytes
ASM      CMD    77 Recs      10 Kbytes
TOD      CMD    21 Recs       3 Kbytes
COPYDISK CMD    19 Recs       3 Kbytes
DDT86    CMD   109 Recs      14 Kbytes
ASSIGN   CMD    16 Recs       2 Kbytes
NEWDISK  CMD    50 Recs       7 Kbytes
STAT     CMD    73 Recs      10 Kbytes
GENCMD   CMD    45 Recs       6 Kbytes
[illegible] CMD [illegible]   1 Kbytes
DUMPDISK LST     8 Recs       1 Kbytes
DUMPDISK H86    18 Recs       3 Kbytes

Press any key to continue.

DUMPDISK SYM     5 Recs       1 Kbytes
DUMPDISK CMD    10 Recs       2 Kbytes
DUMPDISK A86    36 Recs       5 Kbytes
DUMPDISK BAK    37 Recs       5 Kbytes

The Size of 24 files is 129 KBytes
```

(3) CAT

이 프로그램은 DOS 상에서 TYPE 명령어 같은 기능을 갖는 것으로 CP/M-86의 텍스트 화일을 지정된 디바이스로 출력시키는 화일이다. 만약 디바이스의 지정이 없으면 모니터상으로 출력이 되고, 다음과 같은 형식을 요구한다.(화면 3 참조)

CAT〔드라이브명〕화일명〔디바이스명〕

CP/M-86 디렉토리 구조

(1) CP/M-86 디스크상에서의 화일위치

MS-DOS 상에서의 화일관리는 FAT(File Allocation Table) 영역에 기록된 클러스터(Cluster : 디스크 양면의 1섹터씩 합친 것을 말함)를 통해서 관리하고, CP/M-86에서는 디렉토리 엔트리(Directory Entry : 1트랙의 1~4섹터영역)에 기록된 블록(Block) 이라는 개념으로 화일을 관리한다는 것을 지난호 CP/M-86의 분석에서 언급한 바가 있다.

즉, CP/M-86에서는 2의 BSF(Block Shift Factor : 데이터 저장 단위) 승에 1레코드(128바이트)를 곱한 수(2^4*128=2048=2K)가 하나의 블록이 되고, 이를 디스크에 저장되는 최소단위로 하여 기록이 이루어지고 있다. 따라서 CP/M디스크의 FCB(File Control Block)는 이러한 블록 정보를 화면 4(지난호 'CP/M-86의 분석' 148페이지 참조)와 같은 형태로 보관하고 있다. 여기서 하나의 블록은 2048바이트(2K바이트)가 되며 1트랙은 8섹터, 1섹터는 512바이트로 정해져 있으므로 1블록은 1/2트랙과 일치하고, 4물리섹터와 같게 된다.

그리고 여기서 주의할 점은 이 블록

〔화면 3〕 텍스트 화일을 화면상에 출력

```
C:\SYJ>CAT DUMPDISK.A86

;
;       DISK DUMP PROGRAM
;
;               PROGRAMMED BY KIM,JONGSEOK
;

BDOS    EQU     0E0H    ; BDOS INTERRUPT
SELDSK  EQU     09H     ; SELECT DISK FUNCTION OF BIOS
SETTRK  EQU     0AH     ; SET TRACK NO FUNCTION OF BIOS
SETSEC  EQU     0BH     ; SET SECTOR
;

                DB      0DH,0AH,'    JUN 13,1989',0DH,0AH,'$'
;
PROMPT          DB      0DH,0AH,'D)ISK   * : EXIT   LEFT/RIGHT : SECTOR   UP/DN
: TRACK   HOME/END$'
ERRMSG          DB      'DISK IS UNABLE TO READ THIS SECTOR',0DH,0AH,'$'
;
DISKID          DB      0DH,0AH,'      DISK : $'
TRACKID         DB      '       TRACK : $'
SECTORID        DB      '       SECTOR : $'
HEADER          DB      '-----------------------------------------------------
-----------------',0DH,0AH
                DB      '0000  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F',
0DH,0AH
                DB      '-----------------------------------------------------
-----------------',0DH,0AH,'$'
```

〔화면 4〕 FCB의 내용

```
     DISK : 01        TRACK : 01        SECTOR : 01
-----------------------------------------------------------------------
0000  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
-----------------------------------------------------------------------
0000  00 44 44 54 38 36 20 20 20 43 4D 44 00 00 00[6D]  .DDT86  CMD...m
0010 [19 1A 1B 1C 1D 1E 1F]00 00 00 00 00 00 00 00 00  ................
0020  00 41 53 53 49 47 4E 20 20 43 4D 44 00 00 00 10  .ASSIGN  CMD....
0030  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...............
0040  00 4E 45 57 44 49 53 4B 20 43 4D 44 00 00 00 32  .NEWDISK CMD...2
0050  21 22 23 24 00 00 00 00 00 00 00 00 00 00 00 00  !"#$............
0060  00 46 55 4E 43 54 49 4F 4E 43 4D 44 00 00 00 0E  .FUNCTIONCMD....
0070  25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  %...............

D)ISK  * : EXIT   LEFT/RIGHT : SECTOR   UP/DN : TRACK   HOME/END
```

이라는 개념이 크기(SIZE)의 개념이 아니라, 위치의 개념이므로 트랙과 섹터를 블록이라는 변수로 일원화하였다고 생각하면 틀림없다. 따라서 이런면에 있어서는 DOS의 클러스터와 같은 개념으로 파악해도 무방하다. 그리고 클러스터가 연결된 리스트(LINKED LIST)의 구조를 갖고 계속되는 화일의 위치를 기록하는데 반해서 CP/M은 일종의 어레이(ARRAY) 구조를 통해서 화일의 위치를 기록한다는 차이점이 있다. 링크된 화일은 리스트가 어떤곳에 있든지간에 반드시 다음의 위치를 알고 있어야 한다. 이것은 꼬리에 꼬리를 연결지어놓은 것같은 구조를 하고 있으며, 어레이구조는 최소 단위인 블록으로 구분지어져 있는데 이 블록의 수는 모두 180(360K/2K)개로 이루어져 있다(**그림 1** 참조).

다음은 화면 4의 디렉토리 목록을 자세히 살펴보자. 이 그림에 대한 설명은 지난호에서도 잠시 언급하였으므로, 블록과 관련된 부분만을 설명하기로 하겠다. 이 그림에서 00H번지부터 0FH번지까지는 'DDT86. CMD'라는 화일 이름이 있고 다음 10H번지에서 1FH 번지까지는 이 화일에 대한 디스크상의 위치를 블록 단위로 표시하고 있다.

따라서 디스크상에서의 'DDT86. CMD' 화일의 위치, 즉 크기는 19H블록에서부터 1FH블록에 이르는 7개의 블록으로 구성되어 있으며, 이로부터 이 화일이 디스크상에서 차지하는 부분이 7×2K=14K 바이트임을 알 수 있다.

다음 ASSIGN.CMP의 화일크기는 30H에서 보듯이 1개 블록(20H)을 차지하고 있다.

그러나 실제 화일 크기와 일치하지 않을 수도 있다. 왜냐하면, 실제 화일의 크기는 0FH번지에 있는 6DH (=109) 개의 레코드로 이루어져 있고, 1레코드당 128바이트로 구성되어 있다고 했으므로 이는 109×128바이트 즉,13.6 K바이트이기 때문이다.

그러면 이제 이 블록과 트랙, 섹터는 어떤 관계를 갖고 있는지 알아보자. 먼저 트랙 0은 CP/M-86의 시스템이 차지하는 영역이고, 디렉토리 엔트리는 트랙 1, 물리 섹터 1에서 4까지에 기록되어 있으므로, 첫번째 데이터가 기록될 수 있는 디스크에서의 위치는 트랙 1, 물리 섹터 5~8이 된다. 따라서 첫번째 블록은 트랙 1, 물리 섹터 5~8이 되며, 두번째 블록은 트랙 2, 물리 섹터 1~4가 된다(**그림 2** 참조). 이런 식으로 서로의 관계를 맺고 있으므로 블록과 트랙, 섹터간에는 다음과 같은 관계식이 성립한다.

블록=(트랙−1)*2+섹터 DIV 5
(단, 트랙>0)
트랙=블록 DIV 2+1
섹터=(블록 MOD 2)*4+1

한편, CP/M-86의 양면 디스크의 경우 논리적인 트랙 구조가 그림 2(b), (c)와 같으므로, 논리 트랙이 39트랙을 넘을 경우 물리 트랙은

물리 트랙=79−논리 트랙(SIDE=1)

이 된다.

다시 화면 4를 보면서 예를 들어보자. 'DDT86.CMD'의 시작 블록은 10H번지에 있는 19H(=25)로 이것의 논리 트랙을 구해보면

트랙=25DIV 2+1=13트랙(SIDE=0)

이고,

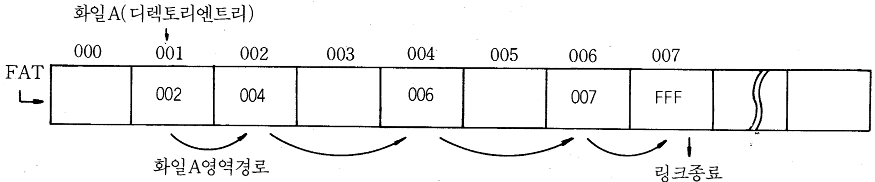
섹터=(25 MOD 2)*4+1=5

섹터이다.

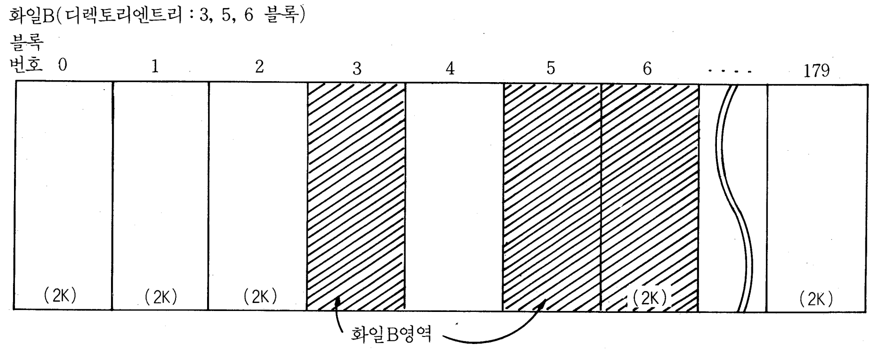
따라서 디스크상에서 'DDT86.CMD'의 시작 위치는 13(=0DH)트랙, 5섹터이고 헤드(SIDE)는 0이다. 이와같은 방법으로 계속 나머지 블록에 대해서도 적용하여 계산하면, 이 화일이 속한 트랙, 섹터, 헤드를 찾을 수 있게 된다.

물론 이와 같은 작업은 CP/M-86이 알아서 해주게 되지만, 전송을 위해서는 이러한 작업을 일일이 지정해 주지 않으면 안된다. 만일 CP/M-86 하에서 반대로 이와 같은 전송을 행하고자 한다면, 역으로 MS-DOS의 화일 구조

〔그림 1〕 디스크상의 화일영역

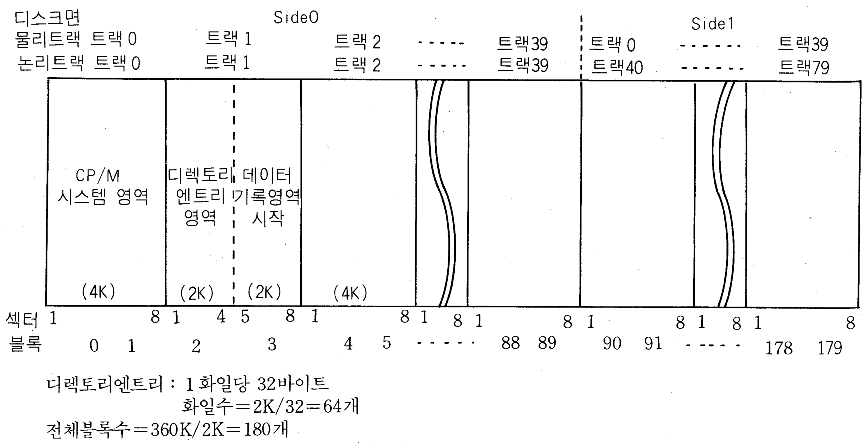


(a) MS-DOS의 Linked List

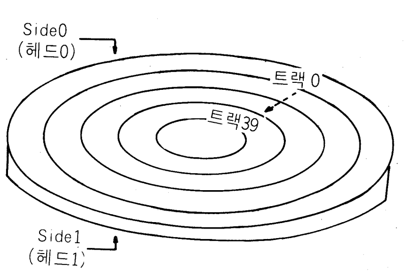


(b) CP/M-86의 화일영역

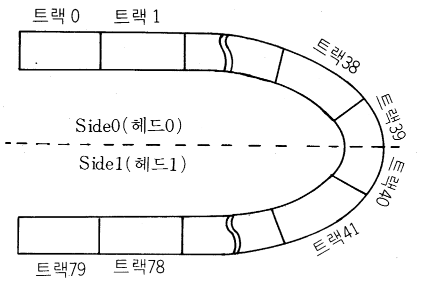
〔그림 2〕 CP/M 디스크의 구조



(a) 디스크의 구성



(b) 디스크의 물리트랙



(c) 디스크의 논리트랙

를 분석하여 DOS가 해주는 작업을 하나 하나 해주어야 한다. 이 경우 MS-DOS의 화일 관리가 CP/M보다 훨씬 복잡하므로 프로그램의 작성이 어렵게 된다.

지금까지 설명한 CP/M의 디렉토리를 이해한 독자라면, 그 다음 전송 부분은 짐작이 갈 것이다. 즉, DOS상에서 원하는 CP/M 화일을 CP/M 디렉토리 영역에서 찾아 일치하는 FCB를 저장한 다음, 그 화일의 블록 위치를 절대 트랙, 절대 섹터로 변환하여 해당하는 디스크 위치에서 데이터를 읽어 DOS 화일로 저장하는 알고리즘을 연상할 수 있을 것이다.

(2) 화일의 등록

지금까지는 CP/M 화일을 DOS 화일로 전송할 때의 메카니즘을 기술하였다. 이제는 DOS 화일을 CP/M 화일로 전송하는 방법을 살펴보자. 이것은 전술한 CP/M 화일을 DOS 화일로의 전송을 이해한 독자라면, 그리 어려운 일이 아님을 알 수 있을 것이다. 이 전송의 핵심은 전송할 CP/M디스크의 빈 디렉토리 영역을 찾는 것이다. 따라서 먼저 해야할 것은 CP/M 디스크의 디렉토리 영역을 읽어들여 빈 디렉토리 영역이 있는지를 살펴보고, 다음에 빈 블록을 찾아서 그 디렉토리 엔트리에 그 블록을 써 넣음과 동시에 해당 블록에 DOS에서 읽어들인 화일을 써 넣으면 되는 것이다.

그러면 빈 디렉토리 엔트리는 어떻게 찾는가? 그것은 FCB의 제일 첫 바이트가 따라오는 화일에 대한 속성을 나타내는 플래그이므로 그것을 체크하면 된다. 즉, 그 디렉토리 엔트리가 빈 영역이라면, 첫번째 바이트가 E5H 값을 가지고 있어 이것을 살펴보는 것으로 빈 영역인지 아닌지를 알 수 있는 것이다.

이 표시는 처음부터 디렉토리 엔트리가 만들어진 적이 없거나 만들어 졌어도 지워진 화일임을 나타내는 것이기도 하다. 따라서 지워진 화일을 복구시킬 때 단지 이 첫번째 바이트를 00H로 만들어 주는 것만으로 가볍게 복구시킬 수가 있는 것이다.

다음으로 빈 블록을 사용하기 위해서는 사용된 블록을 모두 체크하여 사용하지 않은 블록중 번호가 낮은 순서부터 차례로 사용하면 되는 것이다. 이 말은 360K 양면 디스크의 경우 총 블록이 (360K-6K)/2K=177 블록이므로 이중 사용된 블록을 제외하면 된다는 것이다(여기서 6K는 그림 2에서 보여진 것처럼 CP/M시스템과 디렉토리 엔트리 영역이다.).

프로그램설명 (TRANSFER. PAS)

●PROCEDURE ReadDisk

이 부분은 IBM ROM BIOS의 인터럽트 13H(플로피디스크에 관한 내용포함)를 이용하여 절대 트랙, 절대 섹터를 읽어들이는 부분으로, Turbo Pascal에서 제공하는 Intr 프로시듀어를 사용하였다. 이러한 디스크 액세스 방법을 택한 것은 DOS와 CP/M간의 트랙, 섹터 구조가 달라 다른 방법으로는 디스크가 제대로 액세스되지 않기 때문이다.

실제로 PC Tools와 같은 디스크 유틸리티도 CP/M 디스크를 액세스하지 못한다(이것은 FAT영역을 먼저 검색하기 때문인데 CP/M인 경우 그 구조가 달라 읽어들이지를 못한다).

INT 13H에 관한 전달 인수 및 기능에 관한 것은 'IBM Technical Reference Manual'이나 본지 1월호에 게재된 'IBM PC ROM BIOS 분석'을 참조하기 바란다.

●FUNCTION Search_CPM_File

주어진 CP/M 화일 이름을 디스크상에서 찾는 부분이다.

●PROCEDURE CPM_To_DOS

FCB의 블록 위치를 트랙, 섹터로 전환하고 해당 부분을 ReadDisk를 통해서 읽어들인 다음 DOS 화일로 저장하는 부분이다.

●PROCEDURE GetParameter

명령어 라인에서 전달된 인수 (옵션스위치)를 해석하고 부적절한 인수이면 에러 메시지를 출력하는 부분이다.

기타 다른 프로그램은 TRANSFER에서 사용한 방법과 유사하므로 생략하기로 한다.

맺음말

원래 필자의 생각은 이러한CP/M과 DOS 상의 전송 유틸리티를 CP/M 환경하에서 작성하려 하였으나, 전술한 바와 같이 복잡함과 이왕에 CP/M-86의 구조를 연구한 바에는 DOS 환경하에서의 CP/M 화일에 대하여다루는 것이 의미가 있다고 생각하여 이 방법을 택하게 되었다. 그 이유는 CP/M환경하에서 DOS화일을 전송하고 액세스하는 것은 오히려 CP/M을 연구한다기보다 DOS를 연구하는 것과 마찬가지가 되기 때문이다. 즉, DOS화일을CP/M으로 전송하려면 DOS가 해 주는 일을 프로그램으로 모두 기술해야 하므로 CP/M의 구조에 대해서는 알 필요가 없어지기 때문이다.

짧은 글이나마 읽어주신 독자들에게 감사드리며, 아무쪼록 이 작은 유틸리티가 CP/M을 공부하는 여러 사람들에게 도움을 주는 프로그램이 되었으면 한다. PC

(리스트 1) TRANSFER.PAS

```
{-----------------------------------------------------------}
{                                                           }
{       Transfer Ver. 1.0                                   }
{       CP/M-86 to DOS Interface                            }
{       DOS to CP/M-86 Interface                            }
{                                                           }
{       Copyright (c) 1989 by Challenge Technologies, Ltd   }
{                       programmed by J.S.KIM               }
{                                                           }
{-----------------------------------------------------------}
PROGRAM Transfer;
{
  Usage:
    Transfer [options] direcTOry-mask [direcTOry-mask]
  Options:
    /D      Transfer CP/M-86 file TO MS-DOS file
    /C      Transfer MS-DOS file TO CP/M-86 file
    /B      Transfer Binary file
    /T      Transfer Text file
  DirecTOry mask:
    Path1, Filename.
}

USES Dos,Crt;
VAR
  {  File names to transfer and be transfered   }
   Path1,Path2 : PathStr;   {  Path1  ->  Path2  }
  DosFile : Text;
  {  CP/M  Directory Entry  File name   }
   FCB_name : String;
  {  Directory  CPM_Directory Entry     }
   CPM_Dir : array[0..127,0..15] of byte;
  row,col : word;
  Ch : Char;
  Block : array[0..15] of byte;
  Binary : boolean;         {  Binary or Text flag  }
  CTOD   : boolean;         {  CP/M to DOS flag     }
  SourceDrive : byte;
  DestDrive : byte;
  BlockBuf : array[0..2047] of byte;
  Directory : array[0..159] of byte;
  DirEntry : array[0..15] of byte;
  Blks : integer;
  EndBlock : integer;
  dta : byte;
  ix : integer;
  Len : byte;
  srow : word;

PROCEDURE ReadDisk(Drive,Head,Track,SecTOr,NoSec:byte);
VAR
  Reg : Registers;
BEGIN
  WITH Reg DO BEGIN
    AH := 2;            {  Read  SecTOr  }
    DL := Drive;        {  Drive No      }
    DH := Head;         {  Head  No      }
    CH := Track;        {  Track No      }
    CL := SecTOr;       {  SecTOr No     }
    AL := NoSec;        {  No of secTOrs }
    ES := Seg(CPM_Dir);
    BX := Ofs(CPM_Dir);
    Intr($13, Reg);
  END;
END;

PROCEDURE WriteDisk(Drive,Head,Track,SecTOr,NoSecTOr : byte; _ES,_BX : word );
VAR
  Reg : Registers;
BEGIN
  with Reg DO BEGIN
    AH := 3;            {  Read  SecTOr  }
    DL := Drive;        {  Drive No      }
    DH := Head;         {  Head  No      }
    CH := Track;        {  Track No      }
    CL := SecTOr;       {  SecTOr No     }
    AL := NoSecTOr;     {  No of secTOrs }
    ES := _ES;
    BX := _BX;
    Intr($13, Reg);
  END;
END;

PROCEDURE ResetDisk(Drive:byte);
VAR
  Reg : Registers;
BEGIN
  with Reg DO BEGIN
    AH := 0;            {  Read  SecTOr  }
    DL := Drive;        {  Drive No      }
    Intr($13, Reg);
  END;
END;

FUNCTION Search_CPM_File : boolean;
VAR
  Search,               { This flag is set when CP/M file exists.      }
  EOD : boolean;        { End Of DiecTOry flag                         }
  Dir_name : string;    { Temporary file name on CP/M direcTOry entry  }
BEGIN
  ResetDisk(SourceDrive);
  ReadDisk(SourceDrive,0,1,1,2);    { Read DirecTOry Entry of CP/M-86  }
```

```
EOD := false;            { Initialize flag & variables      }
Search := false;
row := 0;
WHILE (row<63) and (not EOD) and (not Search) DO BEGIN
  Dir_name := '';
  IF (CPM_Dir[row,0]=$E5) and (CPM_Dir[row,1]=$E5) THEN EOD := true;
  IF CPM_Dir[row,0]<>$E5 THEN BEGIN    {  Non-erase file          }
    FOR col := 1 TO 11 DO            {  Get name from dir entry  }
      Dir_name := Dir_name + Chr(CPM_Dir[row,col]);
    IF FCB_name = Dir_name THEN BEGIN
      Search := true;
      FOR col := 0 TO 15 DO
        Block[col] := CPM_Dir[row+1,col];
      END;
    END;
    row := row + 2;
  END;
  Search_CPM_File := Search;
END;

PROCEDURE printhex( deci : byte );
const
  hexa : array[0..15] of char
      = ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');
BEGIN
  write(hexa[(deci and $F0) shr 4],hexa[deci and $0F]);
END;

PROCEDURE ERRORS;
BEGIN
  WRITELN;
  WRITELN('Usage:');
  WRITELN;
  WRITELN('  Transfer [options] source-file [destination-file]');
  WRITELN;
  WRITELN('Options:');
  WRITELN;
  WRITELN('  /D      Transfer CP/M-86 file TO MS-DOS file');
  WRITELN('  /C      Transfer MS-DOS file TO CP/M-86 file');
  WRITELN('  /B      Transfer Binary file');
  WRITELN('  /T      Transfer Text file');
  WRITELN;
  WRITELN('Ex)');
  WRITELN;
  WRITELN('  B>TRANSFER /D /T  A:DUMP.LST B:\DUMP.DOS');
  WRITELN;
  Halt(1);
END;

PROCEDURE CPM_To_DOS;
VAR
  EndOfFile : boolean;
  {      --*   Parameter FOR Read/Write Disk    *--       }
    Drive,             {  Drive No -> A : 0, B : 1, ...      }
    Track,             {  Phisical Track -> 0 - 39 FOR  2DD  }
    SecTOr,            {  Phisical SecTOr -> 1 - 8           }
    Head,              {  Or Cylinder -> 0 or 1 FOR FDD      }
    NoSec : byte;      {  Number of secTOrs TO read          }
  Blk : byte;          {  Block Index                        }
BEGIN
  IF not Search_CPM_File THEN BEGIN
    WRITELN('CP/M-86 file ',Path1,' not found.');
    Halt(2);
  END;
  Assign(DosFile,Path2);          { Create Dos File TO be transfered }
  {$I-}
  REWRITE(DosFile);
  {$I+}
  IF IOresult<>0 THEN BEGIN
    WRITELN('Disk error.');
    ERRORS;
  END;
  IF Block[0]=0 THEN ENDOfFile := true
               ELSE ENDOfFile := false;
  Blk := 0;
  WHILE not ENDOfFile DO BEGIN
    Head := 0;
    Track := Block[Blk] div 2+1;
    IF Track>39 THEN BEGIN
      Track := 79 - Track;
      Head := 1;
    END;
    IF Block[Blk] mod 2 = 0 THEN SecTOr := 1
                             ELSE SecTOr := 5;
    NoSec := 4;
    ReadDisk(0,Head,Track,SecTOr,NoSec);
    Blk := Blk + 1;
    IF Block[Blk]=0 THEN ENDOfFile := true;
    FOR row := 0 TO 127 DO
      FOR col := 0 TO 15 DO BEGIN
        IF keypressed THEN BEGIN
          Ch := ReadKey;
          IF Ch = ' ' THEN Ch := Readkey;
          IF Ch = ^C THEN BEGIN
            WRITELN;
            WRITELN('Transfering canceled.');
            Halt(0);
          END;
        END;
        WRITE(DosFile,Chr(CPM_Dir[row,col]));
        IF (CPM_Dir[row,col] = $1A) and (not Binary) THEN BEGIN
          WRITELN('CP/M file transfered.');
```

```
        WRITELN;
        Halt(0);
      END;
    END; { of sub FOR }
  END; { of WHILE }
  WRITELN('CP/M file transfered.');
END; { of CPM_To_DOS }

PROCEDURE Search_Empty_Block;
VAR
  EOD : boolean;
  TempDrive : byte;
BEGIN
  TempDrive := SourceDrive;
  SourceDrive := 0;
  IF Search_CPM_File THEN BEGIN
    WRITELN;
    Writeln(^G'File ',Path1,' is already exist on CP/M disk.');
    Writeln('Use another target file name,please.');
    WRITELN;
    HALT(0);
  END;
  SourceDrive := TempDrive;
  ResetDisk(0);
  ReadDisk(0,0,1,1,4);
  EOD := false;
  FOR ix := 0 TO 159 DO DirecTOry[ix] := $E5;
  row := 0;
  srow := 0;
  WHILE (row<128) DO BEGIN
    IF (CPM_Dir[row,0]<>$E5) THEN BEGIN
      FOR col := 0 TO 15 DO
        IF CPM_Dir[row+1,col]<160 THEN DirecTOry[CPM_Dir[row+1,col]] := 0;
    END
    ELSE IF srow=0 THEN srow := row;
    row := row + 2;
  END;
  ix := 1;
  WHILE DirecTOry[ix]=0 DO ix := ix + 1;
  Blks := ix;
  writeln;
  writeln('dir map');
  FOR ix := 0 TO 159 DO BEGIN
    printhex(Directory[ix]);  write(' ');
    IF (ix+1) mod 16 = 0 THEN writeln;
  END;
  write('search row : ');  printhex(srow);
  write('  Start Block : '); printhex(Blks);
  Ch := ReadKey;
  IF Ch=#27 THEN Halt(0);
  row := srow;
  IF row>=63 THEN BEGIN
    writeln('Direcory entry  full.');
    Halt(0);
  END;
  FOR ix := 0 TO 15 DO DirEntry[ix] := 0;
END;

PROCEDURE DOS_To_CPM;
VAR
  {       --*   Parameter FOR Read/Write Disk    *--       }
    Drive,             {  Drive No -> A : 0, B : 1, ...    }
    Track,             {  Phisical Track -> 0 - 39 FOR  2DD }
    SecTOr,            {  Phisical SecTOr -> 1 - 8         }
    Head,              {  Or Cylinder -> 0 or 1 FOR FDD    }
    NoSec : byte;      {  Number of secTOrs TO read        }
    Blk : byte;        {  Block Index                      }

PROCEDURE WRITE_CPM_FILE;
BEGIN
  Drive := 0;
  NoSec := 4;
  Track := Blks div 2 + 1;
  Head := 0;
  IF Track > 39 THEN BEGIN
    Track := 79 - Track;
    Head := 1;
  END;
  IF Blks mod 2 = 0 THEN SecTOr := 1
                    ELSE SecTOr := 5;
  WriteDisk(Drive,Head,Track,SecTOr,NoSec,Seg(BlockBuf),Ofs(BlockBuf));
  DirEntry[col] := Blks; col := col + 1;
  Blks := Blks + 1;
  WHILE DirecTOry[Blks]=0 DO Blks := Blks + 1;
  for ix := 0 to 2047 do BlockBuf[ix] := $F6;
  ix := 0;
END;
BEGIN
  Search_Empty_Block;
  assign(DosFile,Path1);
  reset(DosFile);
  for ix := 0 to 2047 do BlockBuf[ix] := $F6;
  ix := 0;  col := 0;
  WHILE not eof(DosFile) DO BEGIN
    read(DosFile,ch);
    write(ch);
    BlockBuf[ix] := ord(ch);
    ix := ix + 1;
    IF (ix = 2047) THEN WRITE_CPM_FILE;
  END;
  IF ix < 2047 THEN WRITE_CPM_FILE;
  FOR col := 0 TO 15 DO
    CPM_Dir[row,col] := 0;
  FOR col := 0 TO 15 DO
    CPM_Dir[row+1,col] := DirEntry[col];
  FOR col := 1 TO 11 DO
```

```
    CPM_Dir[row,col] := ord(FCB_name[col]);
  col := 0;
  WHILE (DirEntry[col]<>0) DO col := col + 1;
  CPM_Dir[row,15] := col * 16;
  Drive := 0;
  Head := 0;
  Track := 1;
  SecTOr := 1;
  NoSec := 2;
  WriteDisk(0,0,1,1,2,Seg(CPM_Dir),Ofs(CPM_Dir));
  WRITELN('File Transfered.');
END;

PROCEDURE GetParameter;
VAR
  Attr: Word;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  S : STRING;
  I,J : WORD;
BEGIN
  IF ParamCount < 1 THEN BEGIN
    WRITELN('Parameter not found.');
    ERRORS;
  END;
  Path1 := '';
  Path2 := '';
  CTOD := True;
  Binary := false;
  FOR I := 1 TO ParamCount DO BEGIN
    S := ParamStr(I);
    IF S[1] = '/' THEN
      FOR J := 2 TO Length(S) DO
        CASE UpCase(S[J]) OF
          'D': CTOD := True;
          'C': CTOD := False;
          'B': Binary := True;
          'T': Binary := False;
        ELSE
          WriteLn('Invalid option: ', S[J]);
          ERRORS;
        END
    ELSE
      IF Path1 = '' THEN Path1 := S
                    ELSE Path2 := S
  END;
  IF Path1 = '' THEN BEGIN
    WRITELN('Absent source file-name.');
    ERRORS
  END;
  D := Path1;
  IF Pos(':',D)<>0 THEN
    IF CTOD THEN
      IF D[1] in ['A','B'] THEN SourceDrive := Ord(D[1]) - 65
      ELSE BEGIN
        WRITELN('Invalid disk drive parameter.');
        ERRORS;
      END
    ELSE
      IF D[1] in ['A'..'E'] THEN SourceDrive := Ord(D[1]) - 65
      ELSE BEGIN
        WRITELN('Invalid disk drive parameter.');
        ERRORS;
      END
  ELSE
    SourceDrive := 0;
  Path1 := FExpand(Path1);
  FSplit(Path1, D, N, E);
  IF CTOd THEN Path1 := N+E
        ELSE Path1 := D+N+E;
  IF CToD THEN FCB_name := N+copy('        ',1,8-Length(N))+copy(E,2,Length(E));
  IF Path2='' THEN Path2 := N + E
            ELSE BEGIN
              Path2 := FExpand(Path2);
              FSplit(Path2,D,N,E);
              IF CTOD THEN Path2 := D+N+E
                      ELSE Path2 := N+E
            END;
  IF not CToD THEN FCB_name := N + copy('  ',1,8-Length(N))+copy(E,2,Length(E));
  WRITELN;
  IF CTOD THEN
    WRITELN('Transfer CP/M file to DOS file : ',Path1,' -> ',Path2)
  ELSE
    WRITELN('Transfer DOS file to CP/M file : ',Path1,' -> ',Path2);
  WRITELN;
  WRITE('Press any key!');  Ch := ReadKey;
  WRITELN;
  If Ch = #27 THEN Halt(0);
END;
BEGIN               {  Body of Main Program  }
  GetParameter;
  IF CtoD THEN CPM_To_DOS
        ELSE DOS_To_CPM;
END.
```

〔리스트 2〕 LS.PAS

```
program LS;
uses dos,crt;
var
  Reg : Registers;
  Buffer : array[0..63,0..15] of byte;
  row,col : word;
  Ch : Char;
  Drive,Track,Sector,Head : byte;
  EOD : boolean;

procedure printhex( deci : byte );
const
  hexa : array[0..15] of char
       = ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');

begin
  write(hexa[(deci and $F0) shr 4],hexa[deci and $0F]);
end;

procedure ResetDisk(Drive:byte);
VAR
  Reg : Registers;
BEGIN
  with Reg DO BEGIN
    AH := 0;          { Read  SecTOr }
    DL := Drive;      { Drive No     }
    Intr($13, Reg);
  END;
END;

procedure ReadDisk;
begin
  with Reg do begin
    AH := 2;          { Read  Sector }
    DL := Drive;      { Drive No     }
    DH := 0;          { Head  No     }
    CH := 1;          { Track No     }
    CL := Sector;     { Sector No    }
    AL := 2;          { No of sectors }
    ES := Seg(Buffer);
    BX := Ofs(Buffer);
    Intr($13, Reg);
  end;
end;

var
  Size : integer;
  Entry : integer;

begin
  Drive := 0;
  ResetDisk(Drive);
  If ParamCount >= 1
  Then If (ParamStr(1)='A:') or (ParamStr(1)='a:')
        then  Drive := 0
        else If (ParamStr(1)='B:') or (ParamStr(1)='b:')
               then Drive := 1;
  Writeln('Dirve ',Chr(Drive+65),':');
  Writeln;
  Sector := 1;
  ReadDisk;
  EOD := false;
  Size := 0;
  Entry := 0;
  row := 0;
  while (row<63) and (not EOD) do begin
    if (Buffer[row,0]=$E5) and (Buffer[row,1]=$E5) then EOD := true;
    if Buffer[row,0]<>$E5 then begin
      for col := 1 to 8 do
        Write(Chr(Buffer[row,col] and $7F));
      Write('  ');
      for col := 9 to 11 do
        Write(Chr(Buffer[row,col] and $7F));
      Writeln('  ',Buffer[row,15]:4,' Recs  ',
             round(Buffer[row,15]/8+0.4999):5,' Kbytes');
      Size := Size + round( Buffer[row,15] / 8 + 0.499);
      Entry := Entry + 1;
      If Entry=20 Then Begin
        Writeln;
        Writeln('Press any key to continue.');
        Ch := ReadKey;
        If Ch = #27 Then Halt(0);
        Writeln;
      End;
    end;
    row := row + 2;
  end;

  Writeln;
  Writeln('The Size of ',Entry,' files is ',Size,' KBytes');
  Writeln;
end.
```

〔리스트 3〕 CAT.PAS

```
program Cat;
{--------------------------------------------------------}
{                                                        }
{       Cat Ver. 1.0                                     }
{       CP/M file typer                                  }
{                                                        }
{       Copyright (c) 1989 by Challenge Technologies, Ltd }
{                       programmed by J.S.KIM            }
{                                                        }
{--------------------------------------------------------}
uses dos,crt;
var
  Reg : Registers;
  Buffer : array[0..$127,0..15] of byte;
  row,col : word;
  Ch : Char;
  Drive,Track,Sector,Head : byte;
  NoSec : byte;
  EOD : boolean;
  EndOfFile : boolean;

procedure printhex( deci : byte );
const
  hexa : array[0..15] of char
      = ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');
begin
  write(hexa[(deci and $F0) shr 4],hexa[deci and $0F]);
end;

procedure ReadDisk;
begin
  with Reg do begin
    AH := 2;          { Read  Sector  }
    DL := Drive;      { Drive No      }
    DH := Head;       { Head  No      }
    CH := Track;      { Track No      }
    CL := Sector;     { Sector No     }
    AL := NoSec;      { No of sectors }
    ES := Seg(Buffer);
    BX := Ofs(Buffer);
    Intr($13, Reg);
  end;
end;

var
  Size : integer;
  Entry : integer;
  Position : byte;
  Fname,Tname : PathStr;
  Len : byte;
  Search : boolean;
  Block : array[0..15] of byte;
  Blk : byte;
  Attr: Word;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  S : STRING;
  I,J : WORD;

begin
  Drive := 0;
  If ParamCount < 1 then
  begin
    writeln('Invalid ty format');
    writeln(' Usage:  CAT  FILENAME.EXT');
    Halt(0);
  end;
  Fname := ParamStr(1);
  Fname := FExpand(Fname);
  FSplit(Fname,D,N,E);
  Fname := N+copy('        ',1,8-Length(N))+copy(E,2,Length(E));
  Writeln;
  Head := 0;
  NoSec := 2;
  Track := 1;
  Sector := 1;
  ReadDisk;
  EOD := false;
  Search := false;
  Size := 0;
  Entry := 0;
  row := 0;
  while (row<63) and (not EOD) and (not Search) do begin
    Tname := '';
    if (Buffer[row,0]=$E5) and (Buffer[row,1]=$E5) then EOD := true;
    if Buffer[row,0]<>$E5 then begin
      for col := 1 to 11 do
        Tname := Tname + Chr(Buffer[row,col]);

      If Fname = Tname Then Begin
        Search := true;
        for col := 0 to 15 do
          Block[col] := Buffer[row+1,col];
      End;
    end;
    row := row + 2;
  end;

  if Block[0]=0 then EndOfFile := true
              else EndOfFile := false;
  Blk := 0;
```

```
while not EndOfFile Do Begin
  Head := 0;
  Track := Block[Blk] div 2+1;
  if Track>39 then begin
    Track := 79 - Track;
    Head := 1;
  end;
  if Block[Blk] mod 2 = 0 then Sector := 1
                          else Sector := 5;
  NoSec := 4;
  ReadDisk;
  Blk := Blk + 1;
  if Block[Blk]=0 then EndOfFile := true;
```

```
  for row := 0 to 127 do
    for col := 0 to 15 do
      if keypressed then begin
        Ch := ReadKey;
        if Ch = ' ' then Ch := Readkey;
        if Ch = #27 then Halt(0);
      end
      else if Buffer[row,col] = $1A then Halt(0)
      else if Buffer[row,col] = $9 then Write('        ')
      else if Buffer[row,col] and $7F <> 7 then
           write(Chr(Buffer[row,col] and $7F));
end; { of while }
```

2단편집이 가능한

더블 에디터

이미숙 / 상명여대 가정교육학과

처음 베이직을 배우던 시절을벗어나 CP/M을 처음 접했던 시절, 제일 보기 좋고 마음에 든 것이 터보 파스칼이었다. 어떠한 언어보다 쓰기 편한 터보 파스칼 덕분에 워드스타 방식의 에디터는 모두 손에 익힐 수가 있었다. (에디터의 모든 동작을 키보드로 옮겨 놓았기 때문에 매뉴얼없이는 그림의 떡이나 다름없지만…)

한번은 이러한 에디터와는 무관하게 에디터 프로그램을 만들려고 한 적이 있었다. 그 당시는 IBM PC를 접할 기회가 없었고, 또 애플에서는 프로그램을 작성하는 것이 싫어서 대형기종에서 만들었었다. 프로그램 전체를 C로 작성했기 때문에 기계적인 특성만약간 고려하면 IBM PC용으로도 무리없이 변환이 가능하였다.

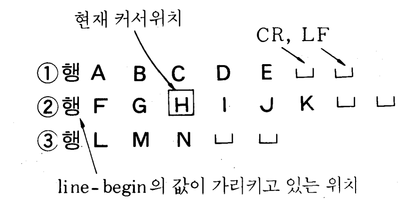
그런데 한가지 문제가 발생하게되었다. IBM PC로 옮긴 프로그램이 이상하게 700줄 이상의 소스는 읽어들이지를 못하는 것이었다. 대형기종에서는 무리가 없었던 것이 왜 탈이 생겼는지 정말 이상한 노릇이었다. 처음에는 필자의 프로그램상 오류인줄만 알고 계속 소스리스트만 체크하고 있었는데 그 이유를 전혀 엉뚱한 곳에서 찾을 수가 있었다. 즉 라인 구조의 문제였던 것이었다. 그 프로그램에서의 구조는 라인을 무조건 80개의 배열로 잡고 각 라인을 포인터로 연결하는 구조였다. 계산을 해보면 (80+2) * 700=57KB로 거의 64KB에 육박하고 있었으니 당연히 메모리가 모자랄 수 밖에 없었다. 16비트는 16비트 포인터를사용하므로64KB가 한계이지만 대형기종은 32비트 포인터를 이용하고 있었으므로 문제가 없었던 것이었다. 여하튼 여러 주변환경 때문에 프로그램 작성에 많은 어려움을 느꼈는데 벌써 이런 추억(?)도 C 언어를 막 시작했던 2년 전의 일이 되고 말았다.

이번의 DOUBLE 에디터는 언젠가 작성하게 될 워드프로세서의 기본 환경으로 만들 목적이었다 (매킨토시의 DTP의 워드프로세서를 보고 생각이 바뀌었지만…). 에디터의 형식은 보통의 에디터와 거의 비슷하나 단순한 하나의 화일을 편집할 수 있는 터보 에디터와는 달리 다단 편집을 시도했다는 점이 보통 에디터와 다른 점이다. 그리고 어휘를 등록할 수 있는 기능이 있어 지루하고 반복적인 단어의 처리를쉽게 했다. 욕심같아서는 블록을 좌우로 시프트하는 문제, 일정구역을 옮기는 문제(Block move와는 다른 개념), 앞에 쓴 문장을 반복해서 써주는 것, 그리고 실행화일이 52KB에 달해서 효율이 그리 좋은 것은 아니므로 RAM상주로 만들어 사이드 킥과 비슷한 작용을 하는 것 등과 같은 여러기능을 첨가하고 싶었지만 시간이 허락하지 않아 실행하지 못했다. 이런 기능들은 후에 기회가 닿는대로 소개할 예정이며 더욱 바람직한 것은 관심있는 독자들이 직접 구현해 보는 일이라 하겠다.

그외 이 프로그램에서 특징있는 몇가지 사항을 요약하면 실제로 사이드킥보다 한 차원이 높은 계산기와 달력이 들어 있으며 2개의 독자적인 화일을 직접 로드해서 작업을 할 수 있고, 서로간에 블록을 복사하거나 이동이 가능하게 만들었다는 점일 것이다. 그리고 이 프로그램은 어셈블리와 C를 혼합해서 작성하였다. C와 어셈블리의 차이는 똑같은 결과를 놓고 볼때 속도에서 약 4배 정도 차이가 난다.

처음 어셈블리 루틴을 혼합해서프로그램을 작성할 때 문제점은 과연 어떤 방법으로 혼합하여 쓸 것인가 하는 것이었다. 매뉴얼을 뒤져봐도 자세한 설명이 나와 있지 않았기 때문에 무척 애를 먹은 적이 있다. 우선 소스프로그

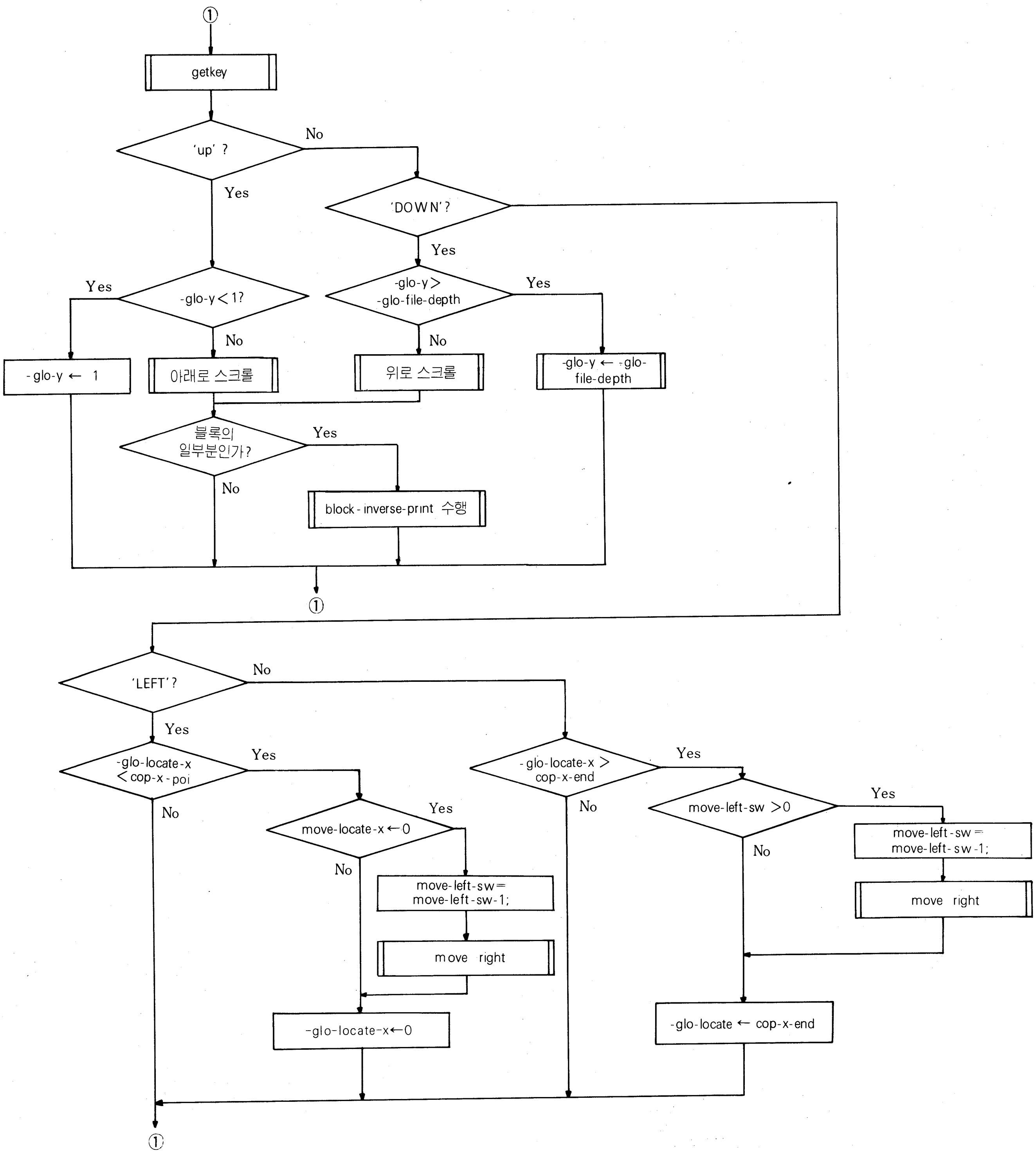
〔그림 1〕 라인의 시작위치와 커서
(CR : Carriage Return LF : Line Feed)



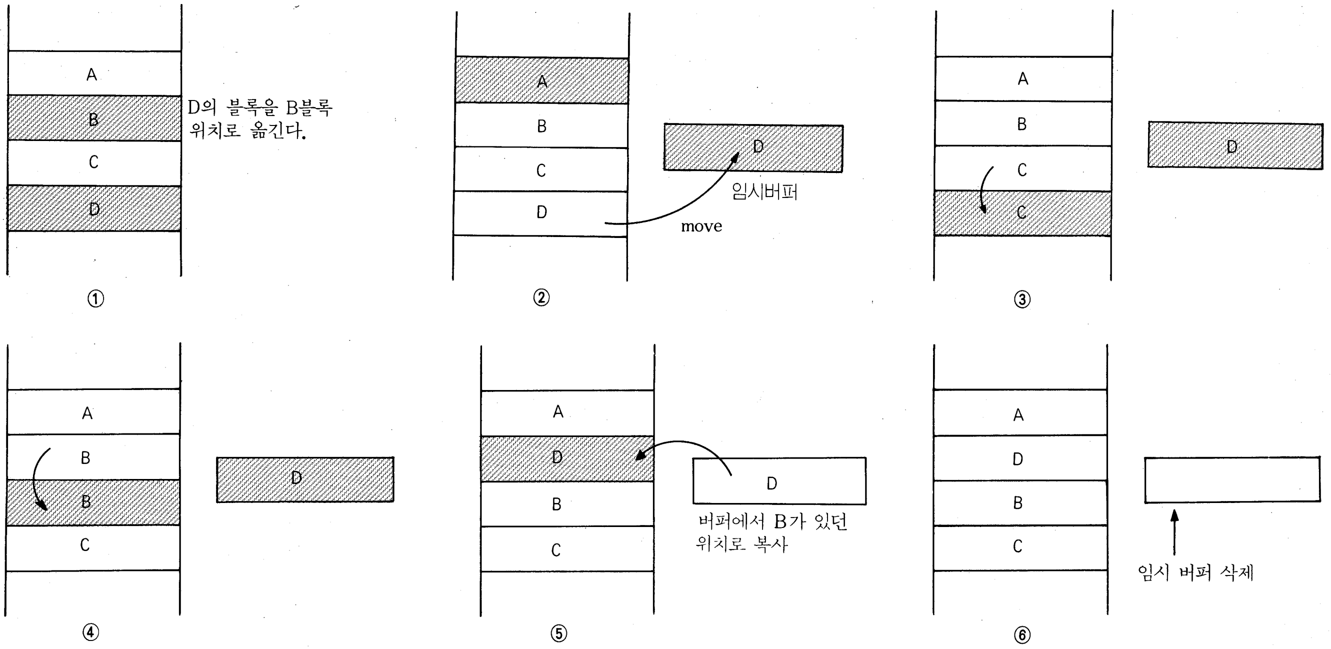
※ leng-line ( ) 에서 ②행의 길이 6을 계산한다.
그리고 CR과 LF의 2개의 문자가 있으므로 2를 더한다.

램 선두에 #pragma inline 을 쓰고 어셈블리가 들어가는 곳에서는 asm을 꼭 첨가해 주고 어셈블리 루틴은 어셈블리 프로그램을 작성하는 방식 그대로 해주면 된다. 이 부분의 자세한 것은 프로그램 리스트를 참조하기 바란다. 그리고 TCC를 실행시킬 때 나오는 옵션들을 보고 메모리 모델과 출력 디렉토리 등을 결정해 주는 것도 잊지 말기

〔그림 2〕 커서의 움직임

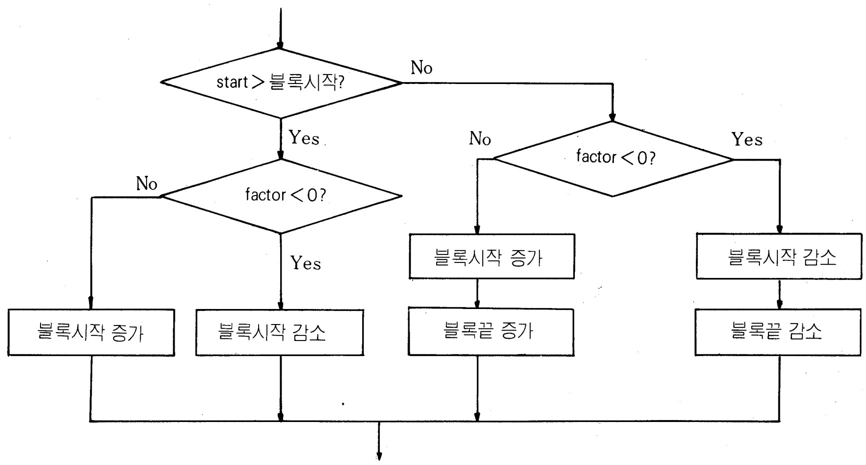


〔그림 3〕 메모리 속에서의 이동



바란다. 플로피디스크 드라이브를 사용하는 독자에게는 다소 무리가 있는 작업일 수도 있다.

〔그림 4〕 끼워넣기와 지우기 흐름도



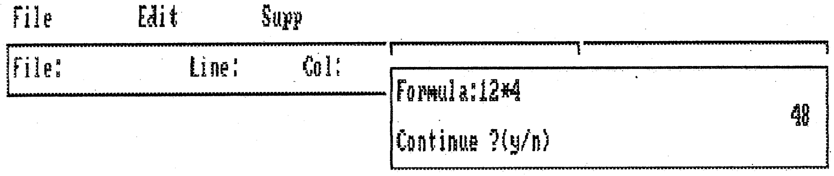
factor : 끼워넣기인가 아니면 지우기인가를 판별
start : 끼워넣기나 지우기가 일어나는 지점

프로그램의 구조

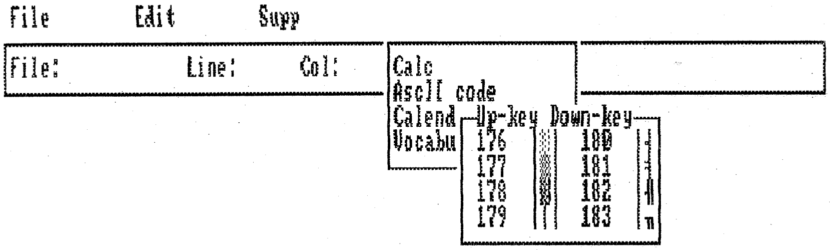
우선 이 프로그램은 실행시에 메인 부분에서 2개의 64KB 용량과 4KB의 비디오 버퍼를 확보한다. 그리고 에디터 방식은, 현재 다루는 화일을 커다란 문자열로 보고 위치와 크기를 정하는 방법을 이용했다. 프로그램의 구조를 잠시 살펴 보면 메인 프로그램인 double.c (**리스트 1** 참조)와 인클루드 화일인 header.h, screen.h, calc.h, asm.h, edit.h로 나누어 진다. double.c는 사용할 모든 기억공간을 확보해 두고 제어권을 select_menu로 넘겨준다. edit.h에는 모든 에디터와 관련한 동작의 루틴을 담고 있으며, header.h에는 모든 변수와 루틴에 대한 선언이, 그리고 screen.h에는 변수의 초기화와 화면의 구성에 관한 내용이 들어 있다. 이같이 한 라인의 크기가 정해져 있는 에디터와는 달리 이 에디터에서는 특이한 방법으로 라인과 컬럼의 위치를 계산한다.

프로그램 중 line_begin이라는 변수는 무부호 변수로써 항상 라인의 시작위치를 지니고 있다. 그리고 leng_line으로는 라인의 길이를 계산한다. 만약 상하 화살표키를 누르게 되면 line_begin과 leng_line에서 계산된 길이에 2를 더해 line_begin의 값을 그 다음 라인의 시작위치로 만든다(**그림 1** 참조). 이 부분은 필자가 처음에 에디터를 작성할 때 생겼던 문제의 해결책이었다. 다른 독자들도 이 테크닉을 생각해서 에디터를 작성해 보기 바란다.

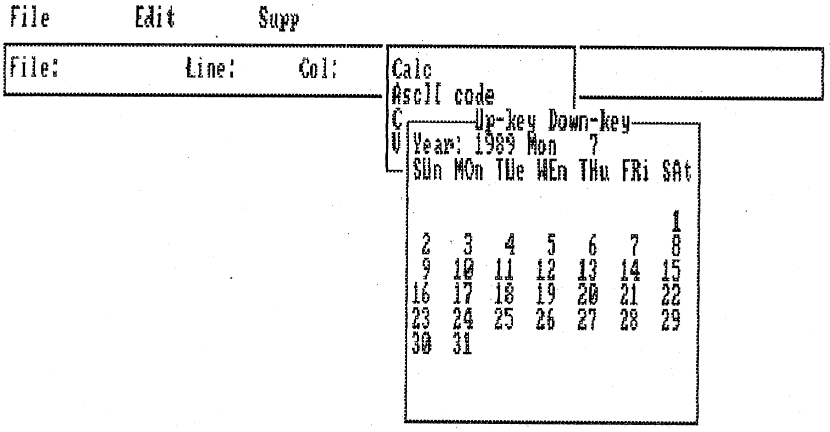
〔화면 1〕 계산기 기능



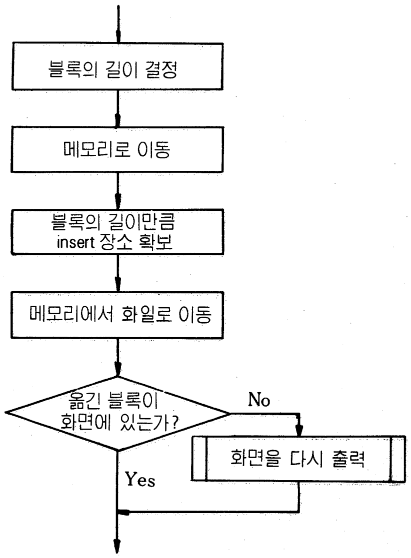
〔화면 2〕 아스키 코드 리스트



〔화면 3〕 달력보기



〔그림 5〕 블록복사



한편 커서의 동작을 이해하기 쉽게 흐름도로 나타내 보면 **그림 2**와 같다.

한 문자의 끼워넣기와 지우기

과연 메모리내에서의 이동기능이 없었으면 터보 에디터가 가능했을까 하는 생각이 든다. 실제로 메모리 이동 기능이 없는 6502에서 터보 에디터는 아마 굼뱅이 버금갈 것이다. 그 일예로 베이직을 길게 짜고 한줄을 첨가시킬 때 리턴을 누르고 기다리는 시간이 조금 길게 느껴지는 것을 들 수 있다. **그림 3**은 메모리속에서 이루어지는 이동 기능을 알기 쉽게 나타낸 것이다. 필자의 프로그램에서는 모든 특수 제어문자들을 제외한 나머지를 배열 line_copy 에 넣고, 입력을 받은후 표준 아스키코드 이외의 제어문자가 들어오면 edit_in_line을 탈출하여 행의 길이의 변화가 있나를 조사하고 그때 비로서 메모리에서 바뀌어야 할 내용을 쓴다. 이 과정을 반복하면서 계속되는 내용을 체크해 나간다.

그리고 블록을 지정하면 무조건 블록명령이 실행되는 것은 아니다. 가령 블록의 시작점에서 끼워넣기가 생기면 블록의 시작점은 insert된 문자만큼 뒤로 밀려야 된다. 그리고 블록의 시작점과 끝점 사이에서 끼워넣기가 일어나면 블록의 끝점만 insert된 수만큼 밀려야 되기 때문이다. 마찬가지로 지우기가 일어나면 블록의 시작과 끝점의 위치가 감소되어야 한다. 이 경우를 순서도로 표시하면 **그림 4**와 같다.

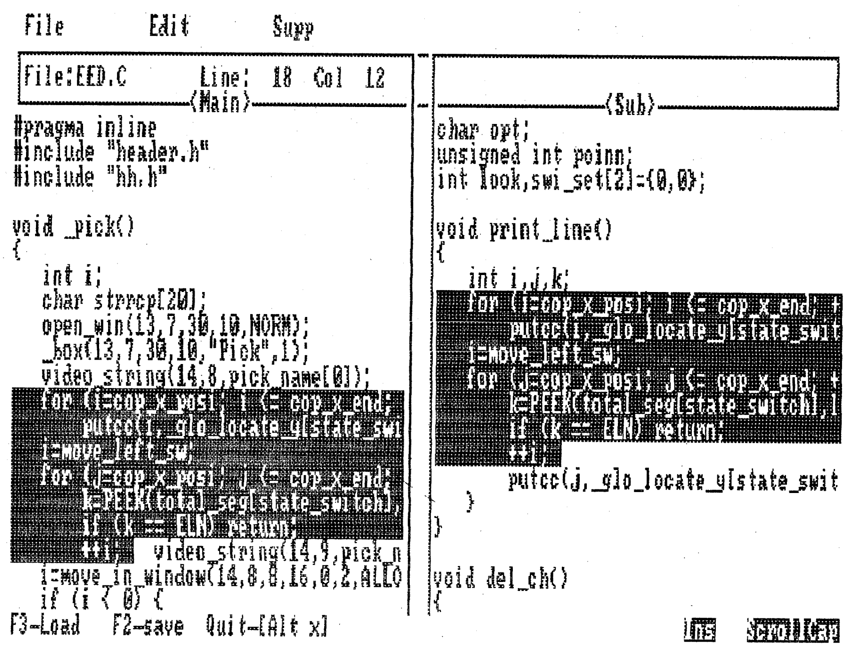
끼워넣기와 지우기는 C의 라이브러리에서 제공해 주는 movedata로 했다. 이는 4KB의 데이터를 기준으로 이동하게 된다. 이제는 앞서 얘기했듯이 Z-80에서 이런 블록이동 명령이 없었다면 터보 에디터의 구현이 어려웠을 거라는 말의 의미를 이해했을 줄로 안다.

광역변수

다음은 DOUBLE 에디터에서 사용한 광역변수들이다.

● **ermo** : 터보 C에서 제공되며 내부의

〔화면 4〕 2 중편집



〔표 1〕 키 기능

키	내 용
←	한 문자 좌측
→	한 문자 우측
↑	한 라인 위로
↓	한 라인 밑으로
Ctrl-F	한 워드 앞으로
Ctrl-N	한 라인 삽입
Ctrl-Y	한 라인 삭제
Ctrl-T	한 워드 삭제
Ctrl-L	find 반복
Alt-X	종료
PgDn	한 페이지 아래로
PgUp	한 페이지 위로
Home	행의 시작점으로
End	행의 끝으로
Ctrl-Q, E	윈도우 맨위로
Ctrl-Q, X	윈도우 맨끝으로
Ctrl-Q, R	화일의 맨위로
Ctrl-Q, C	화일의 맨끝으로
Ins	Insert와 Overwrite 토글
Ctrl-K, K	블록의 끝 지정
Ctrl-K, C	블록 복사
Ctrl-K, Y	블록 삭제
Ctrl-K, V	블록 이동
Ctrl-K, R	블록 읽기
Ctrl-K, W	블록작성을 디스크로
Ctrl-O, I	Autoindent ON/OFF
Ctrl-Q, F	탐색
Ctrl-Q, A	탐색과 대치
F 10	주메뉴 호출
F 2	file save
F 3	file load
↦ ↤	BATCH화일과 ORIGIN 화일 사이를 이동(토글)

소프트웨어적인 에러의 값을 가지고 있다. 함수 _error_print( )에서 에러를 보여 준다.

● **value_getkey** : getkey의 함수에서 얻어지는 키의 값을 받는다. 없으면 NULL을 가진다.

● **_glo_x** : x의 시작위치

● **line_begin** : 각 줄의 시작위치를 항시 갖게 된다.

● **state_switch** : BATCH_ 와 ORIGIN_의 값을 가진다. 현재의 상태가 2중 편집인지 아닌지를 나타내 준다.

● **BORD** : 허큘리스의 비디오 번지 상수

● **get_memory_seg** : require_memory에서 얻어지는 메모리의 시작번지값을 가진다. 나머지 변수는 이름만 봐도 그 의미를 알 수 있기 때문에 생략한다.

프로시듀어별 설명

● 메인 루틴에서

● **getkey( )** : 터보 C에서 제공하는 bioskey( )를 이용하여 입력을 체크하고 입력이 있을 때는 해당값을, 없을 때는 NULL값을 value_getkey에 넘겨준다.

● **_error_print( )** : 소프트웨어적인 에러를 체크해 준다. errno는 터보C에서 제공해 주는 광역변수이다.

● **_load( )** : 만일 '*'을 load_file에서 사용하면 디렉토리 리스트가 나오고 그렇지 않은 경우에는 해당 화일을 로드한다. 중간에 있는 어셈블리 루틴은 리턴값을 체크하여 전체 화일의 행수를 계산한다.

● **_vocabul( )** : 어휘등록을 받는 부분

● **select_menu( )** : file과 edit와 supp의 항목을 선택하는 시작 부분이다.

● **line_edit_in_win** : 윈도우 내에서 행을 입력받을 때 쓰인다.

-**left_posi** : 입력시작 위치

-**right_posi** : 입력의 끝 위치

-**space_allow** : 키입력시 공백의 입력 여부 결정

● **wait_getkey( )** value_getkey가 NULL이 아닐 때까지 기다린다.

● **move_in_window( )** : 윈도우 안에서의 의 커서의 움직임을 처리한다.

-**x,y** : 시작 (x, y) 좌표

-**advance** : →, ←의 키를 누를 때 증감되는 커서의 위치 조정

-**depth** : 최대 x, y의 갯수

-**allow_mode** : 좌우인지, 위아래인지, 또는 상하좌우인지를 판단

● **step** : 동작을 한번만 할 것인지 아니면 여러번 할 것인지를 결정, 나머지 부분은 단어를 보고 충분히 그 내용을 알 수 있을 것이다.

● edit.h에서

edit.h는 에디터의 기능을 담은 프로시듀어이다.

● **decide_x_end( )** : open_sw가 CLOSE_BAT인지 OPEN_BAT 인지를 체

크한다. state_switch가 ORIGIN_인지 BATCH_인지를 결정한다.

● **decide_y( )** : 페이지의 처음 부분 행값과 마지막 부분의 행값을 계산

● **adv_dec_block_one( )** : 블록의 y위치 변경을 계산

● **adv_dec_block_two( )** : 블록의 시작 위치의 변경

● **page_print( )** : 한 페이지나 특정 행 이후를 표시, 블록일 경우는 반전된 상태로 표시된다.

● **result_block_print( )** : 블록명령 실행 후 거쳐가는 부분

● **block_copy( )** : 우선 해당 블록만큼 블록을 할당하고 복사를 실행한다. 이들의 유기적인 관계를 순서도로 표시하면 그림 5와 같다.

● **block_move( )** : 블록복사와 지우기를 한다.

● **check_block( )** : 블록이 화면내에 있는지를 결정한다.

● **eol( )** : end of line을 처리한다.

● **block_process( )** : 블록의 INV와 NORM을 표시해 준다.

● **run_search( )** : 탐색을 실행한다.

● **edit_in_line( )** : edit( )의 부분에서 디폴트 항목이 호출되는 부분이다.

나머지 일부 화일은 변수선언과 화면 초기에 관한 것들이어서 생략한다.

기능들

우선 메뉴항목에서의 SUPP에 관한 내용을 살펴보면 계산기 기능을 볼 수 있다. 이 계산기에서는 베이직에서와 같은 방법으로 입력하면 된다(화면1참조).

아스키 코드의 리스트 보기에서는 상하 화살표키를 이용하면 된다. (화면 2 참조).

달력 보기에서는 이 프로그램을 대부분 완성했던 1989년 5월을 디폴트로 하였다. 상하 화살표키를 이용하며 사이드 킥의 기능에서 참조한 것이다(화면 3 참조).

어휘 등록에서는 10개의 어휘를 40자 이내에서 입력한다. 모두 입력하면 리턴키를 누르고, 〈ESC〉를 누르면 어휘등록을 빠져나오게 된다.

2중 편집에서는 〈ctrl〉-b-j키로 작업을 선언하는데 BATCH화일과 ORIGIN 화일 사이를 탭키로 왔다 갔다 할 수 있다(토글방식). 그리고 화일을 로드할 때는 탭키를 누른 상태에서〈F3〉을 눌러 화일이름을 입력하면 해당 윈도우에 화일이 로드된다. 여기서는 각각의 윈도우내에서 별개의 화일을 편집할 수도 있고 양쪽 화일끼리 내용을 복사할 수도 있게 된다(화면 4 참조). 이 부분은 터보 에디터의 단점이었던 짜집기, 즉 여러 화일에서 일부 내용을 불러올 때의 단점을 해결한 것이다.

그리고 이 프로그램에서 쓰이는 키 기능을 표 1에 나타내었으므로 참고로 하기 바란다.

맺음말

처음으로 TCC를 이용하여 프로그램을 작성했기 때문인지 프로그램 개발이 무척 느린 속도로 진행되었다. 그렇게 느린 컴파일이 끝나고 실행버그를 발견하면 또 수정하고 다시 컴파일을 하는 과정을 눈물겹도록 반복했다. 게다가 컴파일시 용량이 모자라 오브젝트 화일을 만들 수 없다는 메시지가 나오면 플로피디스크 드라이브만을 갖고 있는 필자 자신이 얼마나 원망스러웠는지 모른다. 다행히도 퍼스널컴퓨터의 강영도 기자의 도움으로 이러한 지루한 작업에서 다소 헤어날 수가 있었는데 이 지면을 빌어 고마움을 전하고 싶다. 끝으로 이 프로그램은 필자의 능력이라기보다는 좋은 계절, 8월에 한눈팔지 않은 노력의 결실이라고 보고 싶다. PC

〔리스트 1〕 double. c 소스리스트

```
#pragma inline
#include "header.h"

void getkey()
{
   unsigned int key_s,i;
   value_getkey = NULL;
   check_bios_equip();
   if (bioskey(1) == 0) return ;
   key_s=bioskey(0);
   if ((key_s & 0x00ff) == 0) {
      key_s=key_s >> 8;
      key_s += 256;
      value_getkey=key_s;
      return ;
   }
   value_getkey=key_s & 0x00ff;
}

void ask_save()
{
   error_box("Do you want to file save ?(y/n)");
   if (value_getkey == 'y') goto run_save;
   return ;
run_save:
   _save();
}

#include "asm.h"
#include <dir.h>
#include <process.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void error_box(item)
char *item;
```

```
{
   open_win(30,15,70,17,INV);
   _box(30,15,70,17,"Presse any key",1);
   video_string(31,16,item);
   wait_getkey();
   close_win(30,15,70,17);
}

void _error_print()
{
char *error_list[] = {
"Invalid function number",    "File not found",
"Path not found",             "Memory blocks destroyed",
"invalid memory block address","Invalid environment"
"Invalid format",             "Invalid access code",
"Invalid data",               "Invalid drive specified",
"Attempt to remove CurDir",   "Not same device",
"No more files",              "No such file or directory",
"Too many open files",        "Permission denied",
"Bad file number",            "Not enough core",
"No such device",             "Invalid argument",
"Arg list too long",          "Exec format error",
"Cross-device link",          "Math argument",
"Result too large",           "Disk full ,Retry",
   "Read not ability"};

   if (errno == 0) return ;
   error_box(error_list[errno]);
   errno=0;
}

int require_memory(field)
unsigned int field;
{
   asm mov ah,48h
   asm mov bx,field
   asm int 21h
   asm jc  short fail
   asm mov get_memory_seg,ax
   return 0;
fail:
   return -1;
}

void demish memory(field)
unsigned int field;
{
   asm push es
   asm mov bx,0
   asm mov ah,049h
   asm mov es,field
   asm int 21h
   asm pop es
}

void gotoxy(x,y)
int x,y;
{
   asm mov ah,02h
   asm mov dl,x
   asm mov dh,y
   asm mov bh,0
   asm int 10h
}

void _alternate(x,y,leng)
int x,y,leng;
{
   int i,a,j;
   y=y*160;
   a=PEEK(BORD,y+x*2+1);
   if (a == INV) a=NORM;
   else a=INV;
   for (i=0; i < leng; ++i) {
      asm mov ax,x
      asm add ax,i
      asm shl ax,1
      asm mov j,ax
      POKE(BORD,j+y+1,a);
   }
}

void alternate(x,y,leng,set)
int x,y,leng,set;
{
   int i,a,j;
   y=y*160;
   for (i=0; i < leng; ++i) {
      asm mov ax,x
      asm add ax,i
      asm shl ax,1
      asm mov j,ax
      POKE(BORD,j+y+1,set);
   }
}

void _box(x1,y1,x2,y2,_mes,w)
int x1,y1,x2,y2;
char _mes[];
int w;
{
  int i,j,_glo,len;
  for (i=x1+1 ; i <= x2 ; ++i) {
     putcc(i,y1,196+9*w);
     putcc(i,y2,196+9*w);
  }
  for (i=y1+1 ; i <= y2-1 ; ++i) {
     putcc(x1,i,179+7*w);
     putcc(x2,i,179+7*w);
  }
  putcc(x1,y1,218-17*w);
  putcc(x2,y1,191-4*w);
  putcc(x2,y2,217-29*w);
  putcc(x1,y2,192+8*w);
  if (_mes[0] == '0') return ;
  for (len=0; _mes[len] != '\0'; ++len);
  _glo=len;
  _glo=(x2-x1)/2+x1-_glo/2;
  for (i=0; _mes[i] != '\0'; ++i) putcc(_glo+i,y1,_mes[i]);
}

void check_bios_equip()
{
  int key_res,i,j;
  key_res=bioskey(2);
  if ((key_res & 0x20)) {
```

```
    video_string(68,24,"Num");
    alternate(68,24,3,INV);
  }
  else
  {
    video_string(68,24,"   ");
    alternate(68,24,3,NORM);
  }
  if ((key_res & 0x10)) {
    video_string(71,24,"Scroll");
    alternate(71,24,6,INV);
  }
  else
  {
    video_string(71,24,"      ");
    alternate(71,24,6,NORM);
  }
  if ((key_res & 0x40)) {
    video_string(77,24,"Caps");
    alternate(77,24,4,INV);
  }
  else
  {
      video_string(77,24,"    ");
      alternate(77,24,3,NORM);
   }
   if (key_res & 0x08) {
      Alt_print();
      return ;
   }
}

void echo()
{
   asm mov ah,01h
   asm mov ch,0ch
   asm mov cl,0dh
   asm int 10h
}

void print(x,y,num)
int x,y,num;
{
   gotoxy(x,y);
   printf("%4d",num);
}

void putcc (x,y,character)
int x,y;
char character;
{
   int i,j;
   i =x << 1;
   j =y << 7;
   j += y << 5;
   asm push es
   asm mov es,BORD
   asm mov bx,i
   asm mov ax,j
   asm add bx,ax
   asm mov dl,character
   asm mov es:[bx],dl
   asm pop es
}

void video_string(x,y,string_data)
int x,y;
char string_data[];
{
   int i,j,k;
   char gch;
   i = y << 7;
   j = y << 5;
   y= i + j;
   for (i=0 ; (gch=string_data[i]) != '\0';++i) {
    j = (i+x) << 1;
    POKE(BORD,j+y,gch);
   }
}

void noecho()
{
   asm mov ah,1
   asm mov ch.20h
   asm mov cl,0dh
   asm int 10h
}

#include "screen.h"

void fill_null()
{
   int i;
   for (i=0; i <= 19; ++i)
       line_edit_str[i]='\0';
}

int over_file_ask()
{
   error_box("Overwrite file ?(y/n)");
   if (value_getkey == 'y') {
      close_win(25,4,48,6);
      return SET;
   }
   return OFF;
}

int soon;
char dirlist[100][13];
char ffile[13];

int get_file()
{
   int i=0,j;
   struct ffblk ffblk;
   j=findfirst(path,&ffblk,0);
   strcpy(dirlist[0],ffblk.ff_name);
   while (!j) {
      j=findnext(&ffblk);
      ++i;
      strcpy(dirlist[i],ffblk.ff_name);
   }
   qsort(dirlist,i,13,strcmp);
   soon=i;
   return 0;
}
```

```
void file_list()
{
   int i,j,k;

   k=poin;
   for (i=1; i <= 9; ++i)
       for (j=1; j <= 4; ++j) {
           if (k > soon) break;
           video_string(12+(j-1)*14,8+i,dirlist[k]);
           ++k;
       }
}

void file_sub_print()
{
   poin += 4;
   file_list();
}
int _load(load_condi)
int load_condi;
{
   int view_file_set,j,length_lo,length_hi,drive_set;
   unsigned int k,ll;
   char dirrec[3];
   int i;
   long int xx;
   if (save_con[state_switch] != 0) ask_save();
   save_con[state_switch]=0;
   poin=0;
   fill_null();
   view_file_set=drive_set=OFF;
   if (load_condi == 1) goto restart;
   if (load_condi == 2) goto loadprocess;
   strcpy(line_edit_str,load_str);
   open_win(5,3,25,5,NORM);
   _box(5,3,25,5,"Load Filename",0);
   if (line_edit_in_win(6,24,4,NOT_SPACE,_END) < 0) {
      close_win(5,3,25,5);
      return -1;
   }
   strcpy(load_str,line_edit_str);
   for (i=0; load_str[i] != '\0'; ++i) {
       if (load_str[i] == '*') view_file_set=SET;
       if (load_str[i] == ':') drive_set=SET;
   }
   if (drive_set == SET) {
      dirrec[0]=load_str[0]; dirrec[1]=load_str[1]; }
   if (view_file_set != SET) goto loadprocess;
   dirrec[0]=getdisk()+'A'; dirrec[1]=':';
   strcpy(path,load_str);
   open_win(10,7,66,17,INV);
   _box(10,7,66,17,load_str,0);
restart:
    get_file();
    file_list();
    if (i == 0) {
       video_string(19,11,"Path not found");
       return -1;
    }
    i=((int)soon/4) > 9 ? 9 : soon/4;
    i=move_in_window(12,9,9,13,14,i,ALLOW_ALL,NSTEP);
    if (i < 0) {
       close_win(10,7,66,17);
       close_win(5,3,25,5);
       return -1;
    }
    strcpy(ffname,dirlist[i]);
    if (load_str[0] == '.')
       if (load_str[1] == '.') {
          for (i=40; i > 0; --i) {
              if (load_str[i] == 0x5c) break; }
          load_str[i]='\0';
          goto restart;
       }
    load_str[0]=dirrec[0]; load_str[1]=dirrec[1];
    for (i=0; i <= 20; ++i)
        load_str[i+2]=ffname[i];
loadprocess:
    __open(load_str,O_RDONLY);
    strcpy(_main_file[state_switch],load_str);
    if (errno) {
       _error_print();
       return -1;
    }
    if (pick_name[0][0] == '\0')
       strcpy(pick_name[0],load_str);
    else strcpy(pick_name[1],load_str);
    filelength(handle);
    asm mov length_lo,ax
    asm mov length_hi,dx
    xx=length_lo+length_hi*16*16*16*16;
    if (length_hi > 0 && state_switch == BATCH_) {
       errno=36;
       _error_print();
       close(handle);
       return -1;
    }
    if (length_hi == 0) {
       k=xx;
       READ(total_seg[state_switch],0,k-1);
       _glo_file_leng[state_switch]=k;
       _glo_file_depth[state_switch]=0;
       ll=total_seg[state_switch];
       asm push ds
       asm mov ds,ll
       asm mov bx,0h
       asm mov cx,0h
start:
       asm mov dl,[bx]
       asm inc bx
       asm cmp dl,0dh
       asm jne jumm
       asm inc cx
 jumm:
       asm cmp bx,k
       asm jne start
       asm mov ll,cx
       asm pop ds
       _glo_file_depth[state_switch]=ll+1;
       gotoxy(60,1);
       goto Jump;
    }
    if (state_switch == BATCH_ || open_sw == SET) {
       _error_print();
       goto jum2;
    }
Jump:
```

```
    if (view_file_set == SET) close_win(10,7,66,17);
    close_win(5,3,25,5);
    close(handle);
    return 0;
jum2:
    if (view_file_set == SET) close_win(10,7,66,17);
    close_win(5,3,25,5);
    close(handle);
    return -1;
}

void _save()
{
    handle=creat(_main_file[0],WR);
    WRITE(total_seg[state_switch],0,_glo_file_leng[state_switch]-1);
    close(handle);
}

void _delete_origin()
{
    int i,j=0;
    char direc[20];
    ffname[0]='\0';
    path[0]='A'+getdisk();
    path[1]=':';
    getcurdir(0,direc);
    for (i=0; direc[i] != '\0'; ++i)
        path[i+2]=direc[i];
    for (j=0; ext[j] != '\0'; ++j)
        path[i+j+2]=ext[j];
    open_win(10,7,70,20,INV);
    _box(10,7,70,20,path,0);
    get_file();
    file_list();
    i=(soon/4) > 9 ? 9 : soon/4;
    i=move_in_window(12,9,9,13,14,i,ALLOW_ALL,NSTEP);
    if (i < 0) {
        close_win(10,7,70,20);
        return ;
    }
    strcpy(ffname,dirlist[i]);
    close_win(10,7,70,20);
    if (unlink(ffname) < 0) errno=3;
    else errno=0;
    _error_print();
}

int __open(filename,_mode)
char filename[];
int _mode;
{
    errno=0;
    handle=open(filename,_mode);
    _error_print();
    return handle;
}

void _rename_origin()
{
    int i,j;
    char direc[20];
    ffname[0]='\0';
    path[0]='A'+getdisk();
    path[1]=':';
    getcurdir(0,direc);
    for (i=0; direc[i] != '\0'; ++i)
        path[i+2]=direc[i];
    for (j=0; ext[j] != '\0'; ++j)
        path[i+j+2]=ext[j];
    open_win(10,7,70,20,INV);
    _box(10,7,70,20,path,0);
restart1:
    get_file();
    file_list();
    i=(soon/4) > 9 ? 9 : soon/4;
    i=move_in_window(12,9,9,13,14,i,ALLOW_ALL,NSTEP);
    if (i < 0) {
        close_win(10,7,70,20);
        return ;
    }
    strcpy(load_str,dirlist[i]);
    if (load_str[0] == '.')
        if (load_str[1] == '.') {
            for (j=40; j > 0; --j) {
                if (load_str[j] == 0x5c) break; }
            load_str[j]='\0';
            goto restart1;
        }
    strcpy(old_name,dirlist[i]);
    open_win(4,3,26,5,NORM);
    _box(4,3,26,5,"00",0);
restart:
    fill_null();
    if (line_edit_in_win(5,24,4,NOT_SPACE,_END) < 0) {
        close_win(4,3,26,5);
        close_win(10,7,70,20);
        return ;
    }
    strcpy(ffname,line_edit_str);
    if (rename(old_name,ffname) < 0) errno=2;
    else errno=0;
    close_win(4,3,26,5);
    close_win(10,7,70,20);
    _error_print();
}
void _pick()
{
    int i;
    char strrcp[20];
    open_win(13,7,30,10,NORM);
    _box(13,7,30,10,"Pick",1);
    video_string(14,8,pick_name[0]);
    video_string(14,9,pick_name[1]);
    i=move_in_window(14,8,8,16,0,2,ALLOW_TOP_DOWN,NSTEP);
    if (i < 0) {
        close_win(13,7,30,10);
        return ;
    }
    close_win(13,7,30,10);
    if (pick_name[i][0] == ' ') _load(2);
    else _load(1);
}

void _new()
{
    if (open_sw == OPEN_BAT) {
        if (state_switch == ORIGIN_) scroll(UP_CODE,0,41,4,79,23);
        else scroll(UP_CODE,0,0,4,38,23);
        _init_batch_job_variable();
```

```
    }
    else
    {
       scroll(UP_CODE,0,0,4,78,23);
       _init_variable();
    }
    strcpy(_main_file[state_switch],"Noname");
}

void batch_screen()
{
    int i;
    for (i=3; i <= 23; ++i) {
        putcc(38,i,179);
        putcc(40,i,179);
    }
    video_string(17,3,"<Main>");
    video_string(57,3,"<Sub>");
}

#include <calc.h>

void vocabul()
{
    int i,j;
    open_win(39,2,79,14,NORM);
    _box(39,2,79,14,"00",0);
    i=0;
    for (j=0; j <= 9; ++j)
        video_string(40,j+3,vocabul_char[j]);
    for ( ; ; ) {
        if (
         (j=move_in_window(40,3,i+3,39,0,10,ALLOW_TOP_DOWN,ONESTEP))< 0)
         goto jum2;
        if (i != j) goto jum;
        fill_null();
        strcpy(line_edit_str,vocabul_char[i]);
        if (line_edit_in_win(40,78,i+3,SPACE,_END) < 0) {
           close_win(39,2,79,14);
           return;
        }
        strcpy(vocabul_char[j],line_edit_str);
 jum:
        i=j;
    }
 jum2:
    close_win(39,2,79,14);
 }

 void save_screen()
 {
    int j,i,k,l=0,m,n;
    for (i=0; i <= 24; ++i) {
        n=i << 7;
        m=i << 5;
        n=m+n;
        for (j=0; j <= 79; ++j) {
            m=j << 1;
            k=PEEK(BORD,m+n);
            POKE(screen_seg,l,k);
            ++l;
            k=PEEK(BORD,m+n+1);
            POKE(screen_seg,l,k);
            ++l;
            }
        }
}


void _o_s_shell()
{
    char *pathname;
    echo();
    pathname=getenv("COMSPEC");
    setdisk(*pathname);
    save_screen();
    if (spawnl(P_WAIT,pathname,pathname,NULL) == -1) return ;
    movedata(screen_seg,0,BORD,0,4000);
    noecho();
}

void quit()
{
    if (save_con[state_switch] != 0) {
       handle=__open(_main_file[state_switch],O_RDONLY);
       if (handle != NULL) ask_save();
    }
    scroll(UP_CODE,0,0,0,79,24);
    echo();
    exit(0);
}


void _ascii()
{
    int asc_pg=0,j,i;
    open_win(44,4,63,9,NORM);
    _box(51,4,53,9,"00",0);
    _box(61,4,63,9,"00",0);
    _box(44,4,63,9,"Up-key Down-key",0);
    for ( ; ; ) {
        j=asc_pg;
        for (i=5; i < 9; ++i) {
            video_string(45,i,"    ");
            print(45,i,j+i-5);
            putcc(52,i,j+i-5);
            video_string(55,i,"    ");
            print(55,i,j+i-1);
            putcc(62,i,j+i-1);
        }
        wait_getkey();
        switch (value_getkey) {
              case UP   : asc_pg -= 8;
                          if (asc_pg < 0) asc_pg=0;
                          break;
              case DOWN : asc_pg += 8;
                          if (asc_pg > 255) asc_pg -=8;
                          break;
              case _ESC : close_win(44,4,63,9);
                          return ;
        }
    }
}

struct date {
              int da_year;
              int da_day;
              int da_mon;
            } today;
```

```
void _calendar()
{
   char ss[5];
   int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
   long int sum_day;
   int i,space,x,y;
   open_win(39,4,46,6,NORM);
   _box(39,4,46,6,"Year",0);
   itoa(today.da_year,ss,10);
   strcpy(line_edit_str,ss);
   line_edit_in_win(40,45,5,NOT_SPACE,_END);
   strcpy(ss,line_edit_str);
   today.da_year=atoi(ss);

   fill_null();
   open_win(42,4,49,6,NORM);
   _box(42,4,49,6,"Mon",0);
   itoa(today.da_mon,ss,10);
   strcpy(line_edit_str,ss);
   line_edit_in_win(43,48,5,NOT_SPACE,_END);
   strcpy(ss,line_edit_str);
   today.da_mon=atoi(ss);

   open_win(39,4,67,16,NORM);

   video_string(40,5,"Year:      Mon:  ");
   video_string(40,6,"SUn MOn TUe WEn THu FRi SAt");
   noecho();
   for ( ; ; ) {
      video_string(46,5,"    ");
      print(46,5,today.da_year);
      video_string(55,5,"  ");
      print(54,5,today.da_mon);
      sum_day=(today.da_year-1)*365+(today.da_year-1)/4
            -(today.da_year-1)/100+(today.da_year-1)/400;
      if ((today.da_year % 4) ==0) ++days[1];
      if ((today.da_year % 100) == 0) --days[1];
      if ((today.da_year % 400) == 0) ++days[1];
      for (i=0; i < today.da_mon -1; ++i) sum_day += days[i];
      space=(sum_day + 1) % 7;
      x=space+1; i=0;
      scroll(UP_CODE,0,40,8,66,15);
      for (y=8; y <= 14; ++y) {
         for ( ; x <= 6; ++x) {
            ++i;
            print(38+x*4,y,i);
            if (i >= days[today.da_mon-1]) goto jum;
         }
         x=0;
      }
jum:
      _box(39,4,67,16,"Up-key Down-key",0);
      wait_getkey();
      switch (value_getkey) {
            case   UP : --today.da_mon;
                        if (today.da_mon < 1) {
                           today.da_mon=12;
                           --today.da_year;
                        }
                        break;
            case DOWN : ++today.da_mon;
                        if (today.da_mon > 12) {
                           today.da_mon=1;
                           ++today.da_year;
                        }
                        break;
            case _ESC : close_win(39,4,67,16);
                        close_win(42,4,49,6);
                        close_win(39,4,46,6);
                        return ;
      }
      days[1]=28;
   }
}

_dir()
{
   int i;
   char dirr[30];
   open_win(5,3,25,5,NORM);
   fill_null();
   _box(5,3,25,5,"Directory name",0);
   if (line_edit_in_win(6,24,4,NOT_SPACE,_END) < 0) {
      close_win(5,3,25,5);
      return -1;
   }
   strcpy(dirr,line_edit_str);
   chdir(dirr);
   if (errno != 0) {
      close_win(5,3,25,5);
      _error_print();
      return -1;
   }
   get_file();
   open_win(10,7,70,20,INV);
   _box(10,7,70,20,load_str,0);
   file_list();
   i=((int)soon/4) > 9 ? 9 : soon/4;
   i=move_in_window(12,9,9,0,14,i,ALLOW_ALL,NSTEP);
   if (i < 0) {
      close_win(10,7,70,20);
      close_win(5,3,25,5);
      return -1;
   }
   close_win(10,7,70,20);
   close_win(5,3,25,5);
   return 0;
}

void select_menu()
{
   int sel_res,i,in_sel_res,j=0;
   unsigned int xx,yy;
   noecho();
jum1:
   sel_res=move_in_window
   (1,0,_menu_x_posi*12+1,4,12,2,ALLOW_ONLY_R_L,NSTEP);
   _menu_x_posi= (sel_res < 0) ? _menu_x_posi : sel_res;
   if (j) goto jum33;
jum2:
   switch(sel_res) {
         case 0 : open_win(1,1,16,11,NORM);
                  _box(1,1,16,11,"00",0);
                  for (i=2; i <= 10; ++i)
                     video_string(2,i,_file_str[i-2]);
            jum3 :
                  noecho();
                  if (j) goto jum33;
```

```
        in_sel_res=move_in_window
        (2,2,file_y_posi+2,14,0,8,ALLOW_TOP_DOWN,NSTEP);
        file_y_posi=(in_sel_res < 0) ? file_y_posi : in_sel_res;
jum33:
        j=0;
        switch (in_sel_res) {
            case 0 :if (state_switch == 0)
                        _init_variable();
                    else _init_batch_job_variable();
                    if (_load(0) < 0)  goto jum3;
                    j=0;
                    clear_block_value();
                    close_win(1,1,16,11);
                    video_string(6,2,"            ");
                    video_string(6,2,_main_file[state_switch]);
                    decide_x_end();
                    scroll(UP_CODE,0,cop_x_posi,4,cop_x_end,23);
                    page_print(4,NORM); sel_res=1;
                                      goto jum2;
            case 1 :_save();          goto jum3;
            case 2 :_delete_origin(); goto jum3;
            case 3 :_rename_origin(); goto jum3;
            case 4 :_pick();          break;
            case 5 :close_win(1,1,16,11);
                    _new();
                    sel_res=1;        goto jum2;
            case 6 :_dir();           goto jum3;
            case 7 :_o_s_shell();     goto jum3;
            case 8 :quit();
        }
        close_win(1,1,16,11);
        break;
case 1 : echo();
         i=edit();
         if (i == 1) {
            j=1;
            sel_res=0;
            in_sel_res=0;
            goto jum2;
         }
         else j=0;
         break;
case 2 :open_win(37,1,55,6,NORM);
        _box(37,1,55,6,"00",0);
        for (i=2; i <= 5; ++i)
            video_string(38,i,_supp_str[i-2]);
 jum4 :
        noecho();
        in_sel_res=move_in_window
        (38,2,option_y_posi+2,17,0,4,ALLOW_TOP_DOWN,NSTEP);
        option_y_posi=(in_sel_res < 0) ? option_y_posi : in_sel_res;
        switch (in_sel_res) {
            case 0 :_calc();     goto jum4;
            case 1 :_ascii();    goto jum4;
            case 2 :_calendar(); goto jum4;
            case 3 :vocabul();   goto jum4;
        }
        close_win(37,1,55,6);
        break;
}
noecho();
goto jum1;
}

int line_edit_in_win(left_posi,right_posi,y,space_allow,eend)
int left_posi,right_posi,space_allow,y,eend;
{
   int cur_posi,end_leng,i;
   echo();
   for (i=0; line_edit_str[i] != '\0' && i != 30; ++i) ;
   if (eend == _END) {
      cur_posi=i;
      end_leng=i+1;
      video_string(left_posi,y,line_edit_str);
   }
   else
   {
      cur_posi=0;
      end_leng=1;
      goto jum1;
   }
   gotoxy(left_posi+cur_posi,y);
   wait_getkey();
   if (value_getkey == RET) return 0;
   if (value_getkey >= (' '+space_allow) && value_getkey <= 'z') {
      for (i=0; i <= (right_posi-left_posi); ++i)
         putcc(left_posi+i,y,32);
      cur_posi=0;
      end_leng=0;
      line_edit_str[0]=line_edit_str[1]='\0';
   }
   goto jum;
jum1:
   gotoxy(cur_posi+left_posi,y);
   wait_getkey();
jum:
   switch (value_getkey) {
         case _ESC : return -1;
         case HOME : cur_posi = left_posi;
                     gotoxy(left_posi,y);
                     break;
         case END  : cur_posi = (right_posi-left_posi)+1;
                     gotoxy(right_posi+1,y);
                     break;
         case DEL  : if (!end_leng) break;
                     for (i=cur_posi; i <= end_leng; ++i) {
                        line_edit_str[i]=line_edit_str[i+1];
                        putcc(left_posi+i,y,line_edit_str[i]);
                     }
                     --end_leng;
                     putcc(left_posi+i,y,0);
                     break;
         case   UP :
         case DOWN : return 0;
         case LEFT : --cur_posi;
                     if (cur_posi < 0) ++cur_posi;
                     gotoxy(left_posi+cur_posi,y);
                     break;
         case RIGHT: ++cur_posi;
                     if (cur_posi > end_leng) --cur_posi;
                     gotoxy(left_posi+cur_posi,y);
                     break;
         case RET  : line_edit_str[end_leng]='\0';
                     noecho();
                     return 0;
         case  _bs : --cur_posi;
                     if (cur_posi < 0) {
                        ++cur_posi;
                        goto jum1;
                     }
```

```
                for (i=cur_posi; i <= (end_leng+1); ++i) {
                    line_edit_str[i]=line_edit_str[i+1];
                    putcc(left_posi+i,y,line_edit_str[i]);
                }
                line_edit_str[end_leng]='\0';
                --end_leng;
                break;
      default   : if (value_getkey >= (' ' + space_allow) &&
                    value_getkey <= 'z') {
                    line_edit_str[cur_posi]=value_getkey;
                    for (i=end_leng; i > cur_posi; --i)
                        line_edit_str[i+1]=line_edit_str[i];
                    putcc(left_posi+cur_posi,y,value_getkey);
                    for (i=cur_posi; i < end_leng; ++i)
                        putcc(left_posi+i,y,line_edit_str[i]);
                    ++cur_posi; ++end_leng;
                    if ((cur_posi+left_posi) > right_posi) {
                       asm mov ah,02h
                       asm mov dl,07h
                       asm int 21h
                       value_getkey=HOME;
                       --end_leng;
                       cur_posi=0;
                       goto jum;
                    }
                }
    }
    goto jum1;
}

void open_win(x1,y1,x2,y2,screen_condi)
int x1,y1,x2,y2,screen_condi;
{
   unsigned int i,j,k,l;
   char m;
   for (j=y1; j <= y2; ++j) {
       l=j << 7;
       k=j << 5;
       l=k+l;
       for (i=x1; i <= x2; ++i) {
          k=i << 1;
          m=PEEK(BORD,k+l);
          POKE(BORD,k+l,32);
          _win_buf[_win_buf_end]=m;
          ++_win_buf_end;
          k = k + l + 1;
          m=PEEK(BORD,k);
          _win_buf[_win_buf_end]=m;
          ++_win_buf_end;
          POKE(BORD,k,screen_condi);
       }
   }
}

void close_win(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
   unsigned int i,j,k,l,m;
   char n;
   for (j=y2; j >= y1; --j) {
       l=j << 7;
       k=j << 5;
       l += k;
       for (i=x2; i >= x1; --i) {
          k=i << 1;
          n=_win_buf[--_win_buf_end];
          POKE(BORD,k+l+1,n);
          n=_win_buf[--_win_buf_end];
          POKE(BORD,k+l,n);

       }
   }
}

void wait_getkey()
{
   for ( ; ; ) {
      getkey();
      if (value_getkey) return;
   }
}

int move_in_window(x,y,state_y,leng,advance,depth,allow_mode,step)
int x,y,leng;
int advance,depth,allow_mode,state_y,step;
{
   int _move_x,_move_y,vert,k;
   if (allow_mode == ALLOW_ONLY_R_L) {
      _move_x=state_y;
      _move_y=0;
   }
   else
   {
      _move_x=x; _move_y=state_y;
   }
   vert=0;
   for ( ; ; ) {
ban:
      _alternate(_move_x,_move_y,leng);
      wait_getkey();
      _alternate(_move_x,_move_y,leng);
ban2:
      switch (value_getkey) {
           case _ESC : if (allow_mode != ALLOW_ONLY_R_L) {
                          poin=0;
                          return -1;
                       }
                       break;
           case LEFT : if (allow_mode == ALLOW_TOP_DOWN) return -1;
                       _move_x -= advance;
                       if (_move_x < x) {
                          if (allow_mode == ALLOW_ALL) {
                             value_getkey=UP;
                             _move_x=x+3*advance;
                             goto ban2;
                          }
                          _move_x = x+depth*advance;
                          if (step == ONESTEP) goto exit_switch;
                       }
                       break;
           case RIGHT: if (allow_mode == ALLOW_TOP_DOWN) return -1;
                       _move_x += advance;
                       if (allow_mode == ALLOW_ALL) {
                          k=(_move_x-x)/advance+(_move_y-y)*4;
                          if (k > soon){
                             _move_x -= advance;
                             goto ban;
                          }
```

```
            if (_move_x > (x+3*advance)) {
               _move_x=x;
               value_getkey=DOWN;
               goto ban2;
            }
            goto ban;
         }
         if (_move_x > (x+depth*advance)) {
            _move_x=x;
            if (step == ONESTEP) goto exit_switch;
            else goto ban;
         }
         break;
case UP   : if (allow_mode == ALLOW_ONLY_R_L) break;
         --_move_y;
         --vert;
         if (_move_y < y) {
            if (allow_mode == ALLOW_ALL) {
               if (vert < 0) {
                  vert=0;
                  ++_move_y;
                  poin -= 4;
                  if (poin < 0) {
                     poin += 4;
                     goto ban;
                  }
                  scroll(DOWN_CODE,1,12,9,69,18);
                  file_sub_print();
               }
               goto ban;
            }
            _move_y = y + depth;
            goto ban;
         }
         if (step == ONESTEP) return _move_y-y;
         break;
case DOWN : if (allow_mode == ALLOW_ONLY_R_L) break;
         ++_move_y;
         if (allow_mode == ALLOW_ALL) {
            ++vert;
            if (depth < vert) {
               --vert;
               --_move_y; goto ban;
            }
            if (vert > depth) {
               --_move_y;
               scroll(UP_CODE,1,12,9,69,18);
               file_sub_print();
               goto ban;
            }
            goto ban;
         }
         if ((_move_y-y) > depth)
            if (allow_mode == ALLOW_TOP_DOWN)
               _move_y=y;
            else --_move_y;
         if (step == ONESTEP) return _move_y-y;
         break;
case PG_DO: if (allow_mode == ALLOW_ALL) break;
         poin += 36;
         if (poin > soon) {
            poin = soon -(soon%4) - 33;
            _move_y=_move_y+depth;
            _move_x=x;
                       }
                       file_list();
                       break;
         case PG_UP: if (allow_mode == ALLOW_ALL) break;
                       poin -= 36;
                       if (poin < 0) {
                          poin = 0;
                          _move_y=y;
                          _move_x=x;
                       }
                       file_list();
                       break;
         case RET  : if (allow_mode == ALLOW_ALL)
                          return (_move_y - y)*4+(_move_x-x)/advance;
                       if (allow_mode == ALLOW_ONLY_R_L)
                          return ((int)_move_x/advance);
                       if (allow_mode == ALLOW_TOP_DOWN)
                          return (_move_y - y);
         default   : if (step == ONESTEP) return _move_y-y;
      }
   }
 exit_switch:
    return -1;
 }

 void print_edge(xx,file_xx)
 int xx,file_xx;
 {
    int i,j,l;
    unsigned int cop_file_end;
    unsigned int k;
    k=_glo_page_begin[state_switch];
    cop_file_end=_glo_file_leng[state_switch];
    for (i=4; i < 24; ++i) {
        for (j=1; j < 160; ++j) {
            ++k;
            l=PEEK(total_seg[state_switch],k);
            if (l == ELN) goto br;
            if (cop_file_end == k) return ;
            if (j == file_xx) putcc(xx,i,l);
        }
br:
        ++k;
    }
}

#include "edit.h"

void _search()
{
   sear_con=SEAR;
   view_search();
   run_search();
}

void next()
{
  int i;
  ++_glo_locate_x[state_switch];
  if ((_glo_x[state_switch]+_glo_locate_x[state_switch]) > cop_x_end) {
     _glo_locate_x[state_switch]=cop_x_end-_glo_x[state_switch];
     ++move_left_sw;
     move_screen_to_left(cop_x_posi,cop_x_end);
     i=leng_line(line_begin);
     if (i < (move_left_sw+_glo_locate_x[state_switch])) return ;
```

```
  print_edge(cop_x_end,move_left_sw+cop_x_end-_glo_x[state_switch]+
}
 mem_i=_glo_locate_x[state_switch]+move_left_sw;
}

void back()
{
   --_glo_locate_x[state_switch];
   if (_glo_locate_x[state_switch] < 0) {
      if (move_left_sw <= 0) {
         _glo_locate_x[state_switch]=0;
         move_left_sw=0;
         return ;
      }
      _glo_locate_x[state_switch]=0;
      move_screen_to_right(cop_x_posi,cop_x_end);
      print_edge(cop_x_posi,move_left_sw);
      --move_left_sw;
   }
   mem_i=_glo_locate_x[state_switch]+move_left_sw;
}

void cont_q()
{
   wait_getkey();
   switch (value_getkey) {
    case  5 : win_top();
              break;
    case 24 : win_sub();
              break;
    case 18 : _glo_y[state_switch]=1;
              move_left_sw=0;
              file_top();
              _glo_locate_x[state_switch]=0;
              break;
    case  3 : _glo_y[state_switch]=_glo_file_depth[state_switch];
              move_left_sw=0;
              _glo_locate_x[state_switch]=0;
              file_sub();
              break;
    case  2 : block_begin_posi();
              break;
    case 11 : block_end_posi();
              break;
    case  1 : search_replace();
              break;
    case  6 : _search();
                    break;
          case 25 : eol();
   }
}

void cont_k()
{
   int i,j;
   wait_getkey();
   switch (value_getkey) {
   case  2 :copy_condi = state_switch;
            block_norm_print();
            i=leng_line(line_begin);
            if (i < (_glo_locate_x[state_switch]+move_left_sw)) return;
            _block_begin[state_switch].posi=line_begin;
            _block_begin[state_switch].y=_glo_y[state_switch];
            _block_begin[state_switch].x=
               _glo_locate_x[state_switch]+move_left_sw;
            block_inverse_print();
            break;
    case 11 : if (copy_condi != state_switch) break;
            block_norm_print();
            i=leng_line(line_begin);
            if ((_glo_locate_x[state_switch]+move_left_sw) > i) j=i;
            else j=_glo_locate_x[state_switch]+move_left_sw;
            _block_end[state_switch].posi=line_begin;
            _block_end[state_switch].y=_glo_y[state_switch];
            _block_end[state_switch].x=j;
            if (_block_begin[state_switch].y >
               _block_end[state_switch].y) break;
            if (ask_block_equ() == SET) {
               block_norm_print();
               clear_block_value();
               return ;
            }
            block_inverse_print();
            break;
    case  3 : block_copy(OFF);
            if (state_switch != copy_condi) {
               i=state_switch;
               state_switch=copy_condi;
               clear_block_value();
               state_switch=i;
            }
            break;
    case 25 : block_delete();
            clear_block_value();
            page_print(4,NORM);
            i=back_leng_line(line_begin);
            line_begin -= i;
            break;
    case  8 : hide_block();
            break;
    case 22 : block_move();
            break;
    case 18 : block_read();
            break;
    case 23 : block_write();
            break;
  }
}
int edit()
{
   unsigned int l,k;
   int j,i,mem_ed,oo,mem_posi;
   gotoxy(_glo_locate_x[state_switch]+_glo_x[state_switch],
         _glo_locate_y[state_switch]);
   mem_i=_glo_locate_x[state_switch]+move_left_sw;
   decide_x_end();
juml:
      print(32,2,_glo_locate_x[state_switch]+1+move_left_sw);
      print(23,2,_glo_y[state_switch]);
juml1:
      gotoxy(_glo_locate_x[state_switch]+_glo_x[state_switch],
       _glo_locate_y[state_switch]);
      getkey();
      for (j=0; j < 4; ++j)
         for (i=0; i < 80; ++i)
            POKE(BORD,i*2+160*j+1,NORM);
 jum:
      switch (value_getkey) {
```

```
case LEFT  : back();
             break;
case RIGHT : next();
             break;
case UP    : --_glo_y[state_switch];
             if (_glo_y[state_switch] < 1) {
                _glo_y[state_switch] =1;
                goto juml;
             }
             --_glo_locate_y[state_switch];
             if (_glo_locate_y[state_switch] < 4) {
                _glo_locate_y[state_switch]=4;
                scroll(DOWN_CODE,1,cop_x_posi,4,cop_x_end,23);
                i=back_leng_line(_glo_page_begin[state_switch]);
                _glo_page_begin[state_switch] -= i;
                line_begin=_glo_page_begin[state_switch];
                print_line();
                block_inverse_print();
                goto juml;
             }
             i=back_leng_line(line_begin);
             line_begin -= i;
             break;
 case DOWN : ++_glo_y[state_switch];
             if (_glo_y[state_switch]
                 > _glo_file_depth[state_switch]) {
                 --_glo_y[state_switch];
                 goto juml;
             }
             ++_glo_locate_y[state_switch];
             if (_glo_locate_y[state_switch] > 23) {
                scroll(UP_CODE,1,cop_x_posi,4,cop_x_end,23);
                _glo_locate_y[state_switch]=23;
                i=leng_line(_glo_page_begin[state_switch]);
                _glo_page_begin[state_switch] += (i+2);
                i=leng_line(line_begin);
                line_begin += (i+2);
                print_line();
                block_inverse_print();
                goto juml;
             }
             i=leng_line(line_begin);
             line_begin += (i+2);
             break;
 case INS  : if (_set_ins == SET) {
                _set_ins=OFF;
                video_string(65,24,"Ove");
             }
             else
             {
                _set_ins=SET;
                video_string(65,24,"Ins");
                alternate(65,2,3,NORM);
             }
             break;
 case DEL  : del_ch();    break;
 case HOME : home_posi(); break;
 case END  : end_posi();  break;
 case PG_UP: page_up();   break;
 case PG_DO: page_down(); break;
 case RET  : if (_set_ins != SET) break;
             i=leng_line(line_begin);
             l=line_begin;
             j=mem_i=_glo_locate_x[state_switch]+move_left_sw;
             if (j > i) j=i;
                ins_char(line_begin+j,2,SET);
                POKE(total_seg[state_switch],
                     line_begin+j,ELN);
                POKE(total_seg[state_switch],
                     line_begin+j+1,0x0a);
                move_left_sw=0;
                ++_glo_file_depth[state_switch];
                adv_dec_block_one(1);
                print_line(); ++_glo_y[state_switch];
                if (autoindent == SET) {
                  _autoindent();
            ins_char(line_begin+j+2,_glo_locate_x[state_switch],SET);
                 for(i=0; i < _glo_locate_x[state_switch]; ++i)
                   POKE(total_seg[state_switch],line_begin+i+2+j,32);
                }
        bb_y :
                i=j;
                k=_glo_locate_y[state_switch];
                if (mem_i >= i || _glo_y[state_switch] ==
                  _glo_file_depth[state_switch]) ++k;
                else line_begin = l;
                scroll(UP_CODE,0,cop_x_posi,k,cop_x_end,23);
                page_print(k-1,INV);
                line_begin += (j+2);
                ++_glo_locate_y[state_switch];
                break;
case _bs   : j=_glo_locate_x[state_switch]+move_left_sw;
             if (j == 0) break;
             i=leng_line(line_begin);
             if (i == j) break;
             --_glo_locate_x[state_switch];
             if (_glo_locate_x[state_switch] < 0) {
                move_screen_to_right(cop_x_posi,cop_x_end);
                _glo_locate_x[state_switch]=0;
                --move_left_sw;
             }
             if (j > i) goto juml;
             j=_glo_locate_x[state_switch]+move_left_sw;
             del_character(line_begin+j,line_begin+j+1,SET);
             print_line();
             break;
case _f2   : _save();
             break;
case _f3   : return 1;
case _f10  : return 0;
case  15   : wait_getkey();
             if (value_getkey == 9)
                if (autoindent == SET) autoindent=OFF;
                else autoindent=SET;
             break;
case  12   : run_search();
             if (sear_con == REPL) repl();
             break;
case 2     : if (state_switch == BATCH_) break;
             if (open_sw == OPEN_BAT) break;
             wait_getkey();
             if (value_getkey != 10) break;
             _glo_locate_x[state_switch]=0;
             if (move_left_sw) page_print(4,INV);
             scroll(UP_CODE,0,38,4,79,23);
             move_left_sw=0;
             open_sw=OPEN_BAT;
             batch_screen();
             decide_x_end();
             break;
```

```
case    9 : if (open_sw == CLOSE_BAT) break;
            state_switch=BATCH_;
            video_string(6,2,"            ");
            video_string(6,2,_main_file[state_switch]);
            decide_x_end();
            line_begin=_glo_page_begin[state_switch];
            decide_y();
            for (i=yy.min; i < _glo_y[state_switch]; ++i) {
                l=leng_line(line_begin);
                line_begin += (l+2);
            }
            break;
case    6 : j=leng_line(line_begin);
            l=_glo_locate_x[state_switch]+move_left_sw
             +line_begin;
            if (j<= (_glo_locate_x[state_switch]+move_left_sw)){
               ++_glo_locate_y[state_switch];
               ++_glo_y[state_switch];
               line_begin += (j+2);
               l=line_begin;
               _glo_locate_x[state_switch] = 0;
               move_left_sw=0;
               if (_glo_locate_y[state_switch] > 23) {
                  _glo_locate_y[state_switch]=23;
                  if (move_left_sw !=0) page_print(4,INV);
                  scroll(UP_CODE,1,cop_x_posi,4,cop_x_end,23);
                  print_line();
               }
            }
            i=0;
            oo=OFF;
            k=PEEK(total_seg[state_switch],l);
            if (space_char(k) == SET) {
               ++l; ++i;
            }
            for (j=l; ; ++j) {
                k=PEEK(total_seg[state_switch],j);
                ++i;
                if (k == 32) oo=SET;
                if (oo == SET &&k > 32 && k <= 'z' ||
                    (space_char(k) == SET)) break;
            }
            _glo_locate_x[state_switch] += (i-1);
            if (_glo_locate_x[state_switch] > cop_x_end) {
               i=_glo_locate_x[state_switch]-cop_x_end;
               for (j=0; j <= i; ++j) {
                   move_screen_to_left(cop_x_posi,cop_x_end);
                   print_edge(cop_x_end,cop_x_end+j);
               }
               _glo_locate_x[state_switch]=cop_x_end-
                  _glo_x[state_switch];
               move_left_sw=i;
            }
            break;
case   25 : i=leng_line(line_begin);
            del_character(line_begin,line_begin+i+2,SET);
            scroll(UP_CODE,0,cop_x_posi,
             _glo_locate_y[state_switch],cop_x_end,23);
            page_print(_glo_locate_y[state_switch],INV);
            --_glo_file_depth[state_switch];
            adv_dec_block_one(-1);
            break;
case   14 : ins_char(line_begin,2,SET);
            _glo_locate_x[state_switch]=0;
            if (move_left_sw != 0) page_print(4,INV);
            scroll(DOWN_CODE,1,cop_x_posi,
             _glo_locate_y[state_switch],cop_x_end,23);
         POKE(total_seg[state_switch],line_begin,ELN);
         POKE(total_seg[state_switch],line_begin+1,0x0a);
            ++_glo_file_depth[state_switch];
            adv_dec_block_one(1);
            break;
case   20 : del_word();
            break;
case  271 : state_switch=ORIGIN_;
            video_string(6,2,"            ");
            video_string(6,2,_main_file[state_switch]);
            line_begin=_glo_page_begin[state_switch];
            decide_y();
            for (i=yy.min; i < _glo_y[state_switch]; ++i) {
                l=leng_line(line_begin);
                line_begin += (l+2);
            }
            decide_x_end();
            break;
case  289 : _menu_x_posi=0;   return 0;
case  288 : _menu_x_posi=2;   return 0;
case  376 : entry=0;          goto jum22;
case  377 : entry=1;          goto jum22;
case  378 : entry=2;          goto jum22;
case  379 : entry=3;          goto jum22;
case  380 : entry=4;          goto jum22;
case  381 : entry=5;          goto jum22;
case  382 : entry=6;          goto jum22;
case  383 : entry=7;          goto jum22;
case  384 : entry=8;          goto jum22;
case  385 : entry=9;          goto jum22;
    jum22 : mem_i=_glo_locate_x[state_switch]+move_left_sw;
            for (j=0; ; ++j) {
                if (vocabul_char[entry][j] == '\0') break; }
            ins_char(line_begin+mem_i,j+1,SET);
            _glo_locate_x[state_switch] += j;
            if (_glo_locate_x[state_switch] > cop_x_end) {
               for (i=0; i <= j; ++j) {
                 move_screen_to_left(cop_x_posi,cop_x_end);
                 print_edge(cop_x_posi,cop_x_end+i);
               }
            }
            decide_x_end();
            for (i=0; i <= j; ++i)
                POKE(total_seg[state_switch],line_begin+mem_i+i,
                     vocabul_char[entry][i]);
            print_line();
            break;
case  301 : quit();
case NULL : goto jum11;
case CONT_K:cont_k(); break;
case CONT_Q:cont_q(); break;
default   : if (_set_ins != SET) break;
            decide_x_end();
            if (value_getkey >= ' ' && value_getkey <= 255) {
              mem_posi=_glo_locate_x[state_switch]+move_left_sw;
               oo=leng_line(line_begin);
               if (oo < mem_posi) {
                  j=mem_posi-oo;
                  ins_char(line_begin+oo,j,SET);
                  l=total_seg[state_switch];
                  for (i=0; i < j; ++i)
```

```
                    POKE(1,line_begin+oo+i,32);
                }
                mem_ed=edit_in_line();
                goto cal_end;
            }
            else goto juml;
    cal_end :
            if (mem_ed > oo)
                ins_char(line_begin+mem_posi,mem_ed-oo,SET);
            k=0;
            for (l=line_begin; ; ++l) {
                POKE(total_seg[state_switch],l,line_copy[k]);
                if (line_copy[k] == 0x0a) break;
                ++k;
            }
            goto jum;
        }
        mem_i=_glo_locate_x[state_switch]+move_left_sw;
        block_inverse_print();

    goto juml;
}

/*void dummy_control_key()
{
    asm mov ax,4400h
    asm mov bx,01h
    asm int 21h
    asm mov dh,0
    asm or  dl,020h
    asm mov ax,4401h
    asm int 21h
}*/

main()
{
    int i;
    /*dummy_control_key();*/
    today.da_year=1989;
    today.da_mon=5;
    if (require_memory(5096) < 0) goto juml;
    total_seg[0]=get_memory_seg;
    if (require_memory(5096) < 0) goto juml;
    total_seg[1]=get_memory_seg;
    if (require_memory(250) < 0) goto juml;
    screen_seg=get_memory_seg;
    scroll(UP_CODE,0,0,0,79,24);
    for (i=0; i < 2; ++i) {
        POKE(total_seg[i],0,ELN);
        POKE(total_seg[i],1,0x0a);
    }
    _init_variable();
    _init_screen();
    _init_batch_job_variable();
    strcpy(_main_file[0],"Noname");
    strcpy(pick_name[0],"Noname");
    strcpy(_main_file[1],"Noname");
    alternate(65,24,3,INV);
    video_string(65,24,"Ins");
    select_menu();
juml:
    scroll(UP_CODE,0,0,0,79,24);
    video_string(0,0,"Memory allocate is fail !!");
}
```

(※ 지면관계상 인클루드 화일인 header.h, screen.h, calc.h, edit.h, asm.h는 디스켓으로 제공됩니다.)

□ 디버깅 룸 □

—'89년 7월호 내용중 다음의 내용들이 오기되었기에 바로 정정합니다.

1. 143쪽 표 5
 • LD A, 41H→LD E, 41H
 • MOV AL, 41H→MOV DL, 41H
2. 147쪽 리스트 3
 • START : MOV CL, 3H→START : MOV CL, 32H
3. 150쪽 그림 9
 • 1K단위→1블록단위
4. 205쪽 왼쪽 첫단의 15번째 줄
 • 제어회로가 들어 있는→제어회로가 들어 있어
5. 207쪽 왼쪽 첫단의 13번째줄
 • 콘트롤러를 슬롯에 삽입하여야 한다.
 →코넥터만 연결하면 된다.
6. 207쪽 오른쪽 첫단의 8번째줄
 • 서비스 요구, 라인변경→서비스 요구라인 변경
7. 212쪽 첫단의 6번째줄
 • 되므로 없었는데 확대출력(Doub-
 →리 되므로 구조변경이 쉽다.

사원(경력·신입) 모집

폐사는 ANALOGUE, DIGITAL, H/W, S/W 등
전자기술을 이용하는 화상처리 전문회사로서,
멀티비젼 SYSTEM, IMAGE 칼라 초음파 어군탐지기,
VISUAL INSPECTION & PROCESSOR,
CRT 주식 전광시세판, 초고해상도 칼라 모니터 제품 등을
개발·생산·판매하는 회사로 크지는 않지만
기술적으로는 큰 회사에 결코 뒤지지 않는 알찬 회사로서
제 2 단계 도약에 동참할 분을 찾습니다.

1. 모집요강 :

모집대상 및 부문		인원	자 격	비 고
간부 및 경력사원	기술개발 연구직	○○	• MICRO PROCESSOR & IMAGE PROCESSOR회로 설계 능력 보유자 • ANALOGUE회로 설계 가능하고, MICRO PROCESSOR 이해 가능자 • C. P. U(8. 16, 32BIT) 이해하고 설계 가능한 자	연구소, 설계실, 개발실 책임 경력자 및 유경험자 우대함
	생산, 개발직, 보조직 (신입사원)	○○	• ANALOGUE, TTL 기본회로 이해하고, 현장근무 유경험자 • FINAL TEST 수리 유경험자 • 실무경력은 없지만 학문적으로 자신있고, 이상의 기술에 도 도전하여 최고의 엔지니어가 희망인 자	컴퓨터 생산, 개발 현장근무 경험자
영 업 직		○	• A/V SYSTEM 관련장비 판매 유경험자 • FA SYSTEM 및 관련장비 판매 유경험자	
관 리 직		○	• 총무, 인사, 노무, 회계, 판매 등 중견업체 관리 유경험자로 간부 능력자 경력자	
※ 각 부문 최고의 전문인을 찾으며, 최고의 보수 보장함 ※ 공히 군필자 및 면제자에 한하며, 해외여행에 결격사유가 없는 자				• 고임금 정착

2. 제출서류 : 자필이력서, 자기소개서(전문분야 상세히 기록) 각 1통
3. 전형방법 : 서류전형후 개별통지, 면접 및 실기전형
4. 제출기한 : 1989년 8월 5일～1989년 9월 5일 소인 유효
5. 제출방법 : 우편 이용 우송을 원칙으로 함(방문 및 전화문의 사양함)
6. 제 출 처 : 서울특별시 은평구 응암동 576-21호(총무인사담당)
※ 이력서 상단에 연락처, 희망직종 명기(간부직은 대표친전 명기)

COSMIC COLOR 보문전자

서울시 은평구 응암동 576-21
TEL : 304-5780, 302-3116
FAX : 302-3116

COPY명령의 단점을 해결한 프로그램

FILL.COM

윤일호 / D.T.U.C

플로피디스크에 백업을 할 경우 화일의 크기가 디스켓 1장 이상일 경우 COPY명령 등을 사용하여 복사를 하려면 매우 불편하다.
여기에 소개하는 FILL.COM은 실행화일 크기가 불과 17KB밖에 되지 않으면서 COPY명령의 단점을 해결한 아주 편리한 유틸리티이다.

프로그램 소개

하드디스크에서 플로피로 백업을 할때 하드디스크의 화일이 플로피의 용량보다 클때는 COPY명령을 사용하기가 불편하다.

이런 경우 다음에 소개하는 COPY명령의 단점을 해결한 FILL이란 프로그램을 사용하면 무척 편리하다. 하드디스크의 화일들을 보관용으로 백업을 할 경우에 백업하고자 하는 화일의 크기가 2~3M바이트 정도 될때는 필요한 화일을 하나의 플로피디스크에 모두 백업할 수 없다.

그래서 여러장의 디스크로 나누어 백업할 수 밖에 없게 되는데 이럴때는 일일이 디스켓의 내용을 확인하면서 카피해야 하는 번거로운 작업을 해야만 한다.

필자도 요즘 새로운 프로그램을 테스트 해보려고 하는데, 하드디스크가 모자라서 정리작업을 하려고 PCTOOLS의 최신 버전인 PCSHELL을 사용하고 있다. 이 프로그램에는 백업기능이 새로 첨가되어 있어서 성능이 강력하긴 하지만 프로그램 용량이 너무 크고 사용방법이 복잡하며, 화일을 정리하는 기능이 포함되어 있지 않다.

그래서 해외 BBS를 뒤져보던중 FILL을 발견하고 두근거리는 마음으로 한번 사용하여 보았는데 필자가 원하던 바로 그런 프로그램이었다. FILL은 실행화일의 크기가 17K바이트 정도인데 사용법은 DOS 명령어인 COPY명령과 비슷하여 사용하기 쉽고 성능은 훨씬 강하다.

COPY 명령을 사용하여 복사를 하면 디스크 용량이 꽉 찰 경우 디스크 용량이 부족하다는 에러 메시지를 내고 끝나기 때문에 다시 복사되지 않은 화일을 찾아서 새 디스크에 복사해 주어야 하지만 FILL은 그럴 필요가 없다.

FILL은 화일을 복사할 때 목적 디스크의 남은 용량을 체크하여 복사할 화일을 선택하며 스스로 알아서 화일을 정리하고, 목적 디스크에 복사될 수 있는 만큼만 최대한으로 화일을 선택하여 복사하므로 디스크의 낭비를 최소화한다.

FILL의 사용방법은 아래와 같이 간단하다. 소스 패스는 여러개 지정할 수 있고 맨 마지막 지정된 패스는 목적 패스가 된다.

```
FILL [-Q] SOURCE-PATH [SOURCE-PATH] ........ TARGET-PATH
```

〔화면 1〕 C:\Y\APPLE 디렉토리 내의 화일

```
Volume in drive C is YOON IL HO
 Directory of C:\Y\APPLE

.            <DIR>      6-01-89  11:25p
..           <DIR>      6-01-89  11:25p
DEMO     MAC    22222   4-18-88  10:50a
BASIC    MAC      597   4-18-88   9:31a
C        MAC     1420   4-18-88   9:32a
INIT     MAC     4670   4-18-88   9:32a
INIT     SRC    13155   6-05-89  11:23a
KEYMAP   BAK     8973   6-18-88   2:04a
ME       COM    62814   6-01-89  11:26a
ME       HLP    46861   3-31-88   2:46a
MEC      BAT      201   2-17-88  11:57a
MEMAC    EXE    35472   4-26-88  10:06a
MESYS    MAC    78702   4-27-88  11:48a
PASCAL   MAC     2780   4-18-88   9:59a
ME1      CHN    38060   4-26-88   9:56a
ME2      CHN    53172   4-26-88   9:56a
STATUS   ME       350   1-01-80  12:04a
MECOMP   TMP       34   1-01-80  12:04a
MULTIED             2   1-01-80   2:19a
INIT     BAK    13066   4-01-89   6:27p
MEERR    TMP       82   6-05-89  11:23a
KEYMAP   ME      8973   6-05-89  11:23a
A        DOC     6070   6-03-89   4:43a
      23 File(s)    9404416 bytes free
```

여기서 [−Q] 옵션은 QUIT모드로 지정된다.

QUIT모드로 지정되면 목적디스켓에 원본디스켓의 화일명과 동일한 것이 있을 경우 다시 복사할 것인가라는 질문 메시지를 생략하고 복사해 버린다. 그럼 실제 FILL을 이용하여 화일을 복사해 보자.

예제로서 필자의 하드디스크에 있는 C:\Y\APPLE 디렉토리에 있는 화일(화면 1 참조)을 B드라이브에 모두 복사하여 보자.

이와같이 APPLE 디렉토리에 있는 화일들은 390K 바이트를 넘기때문에 한장의 플로피디스크에는 복사되지 않는다. 그럼 FILL을 사용하여 이 화일들을 B디스크에 실제로 백업해 보자.

```
C>CD Y\APPLE
C>FILL C:\Y\APPLE\*.* B:
```

을 입력하면 백업되는 메시지가 나오면서 화일들을 B디스크로 백업해 준다. ①번과 같은 메시지가 나오면 새 디스크를 넣고 리턴을 입력하면 된다(화면 2 참조). 물론 B에 넣을 디스크는 포맷되어 있어야 한다.

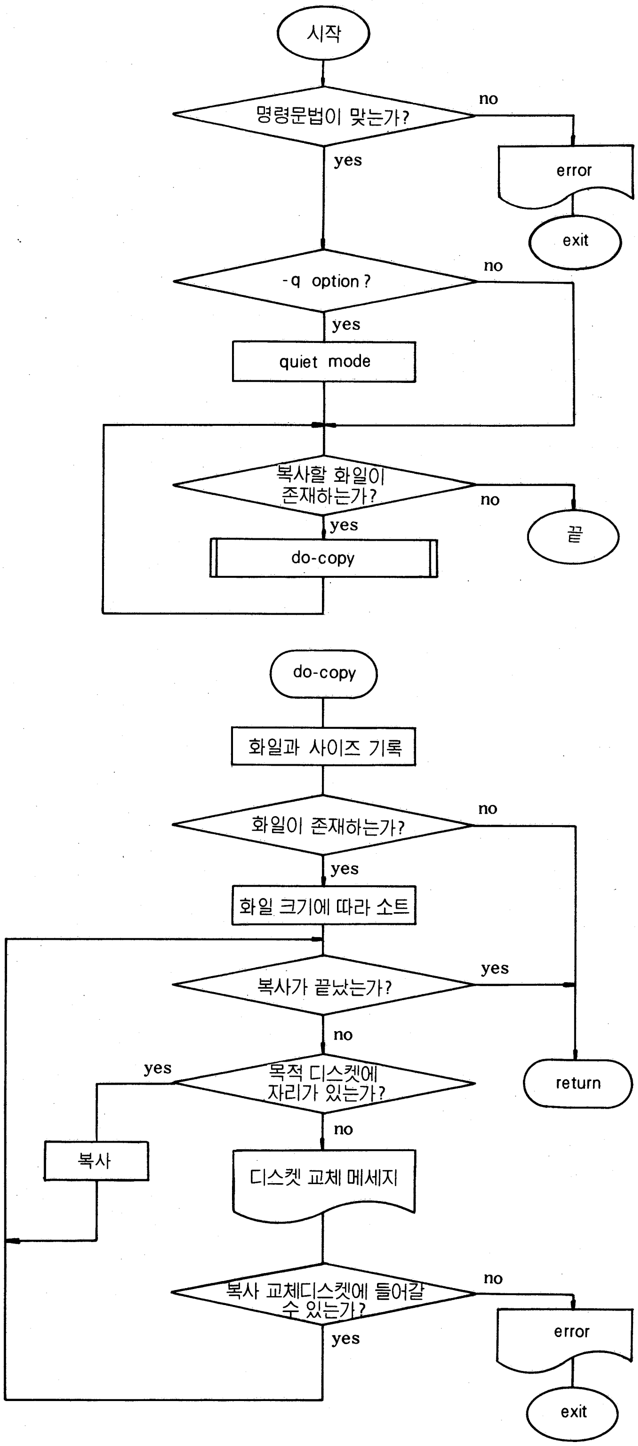
프로그램의 구조

본 프로그램의 소스리스트는 뒤에 수록하였으니 참고하기 바라며 여기에 있는 각 함수중 몇가지 중요한 함수들에 대해 알아보면 다음과 같다. 참고로 프로그램의 전체 알고리즘을 그림 1에 나타낸다.

〔화면 2〕 FILL을 이용한 백업과정

```
362496 bytes remaining. Copying mesys.mac to b:mesys.mac.
283648 bytes remaining. Copying me.com to b:me.com.
220160 bytes remaining. Copying me2.chn to b:me2.chn.
166912 bytes remaining. Copying me.hlp to b:me.hlp.
119809 bytes remaining. Copying me1.chn to me1.chn.
80896 bytes remaining. Copying memac.exe to b:memac.exe.
45056 bytes remaining. Copying demo.mac to b:demo.mac.
22528 bytes remaining. Copying init.src to b:init.src.
9216 bytes remaining. Copying keymap.bak to b:keymap.bak.
Please mount new disk on b:, 12 files (31K) left to copy.
Press Enter to continue. ← 한장이 가득 차면 메시지를 나타내고 복사를 멈춘다.
362496 bytes remaining. Copying init.bak to b:init.bak.
349184 bytes remaining. Copying keymap.me to b:keymap.me.
339968 bytes remaining. Copying init.mac to b:init.mac.
334848 bytes remaining. Copying pascal.mac to b:pascal.mac.
331776 bytes remaining. Copying c.mac to b:c.mac.
329728 bytes remaining. Copying basic.mac to b:basic.mac.
328704 bytes remaining. Copying status.me to b:status.me.
327680 bytes remaining. Copying mec.bat to b:mec.bat.
326656 bytes remaining. Copying meerr.tmp to b:meerr.tmp.
325632 bytes remaining. Copying mecomp.tnp to mecomp.tmp.
324608 bytes remaining. Copying multied to b:multied.
323584 bytes remaining. Copying a.doc to b:a.doc.
```

〔그림 1〕 FILL.COM의 알고리즘



• main(argc, argv)

argc는 이 프로그램을 실행시킬 때 함께 써주는 코맨드 라인상의 파라미터의 갯수이고, argv는 그 파라미터 즉, 문자 포인터 배열에서의 포인터가 된다.

예를 들어 이 프로그램의 이름을 'fill'이라 하면

```
A>fill c:*.* b:
```

리턴을 치면 argc=3이고, argv[0]="fill.exe" 또는 "fill.com"이 된다. 그리고 argv[1]="c:*.*", argv[2]="b:"가 된다.

• do_copy(source, dest)

이 함수는 원본 디스켓에서 화일을 읽어들여 그 이름은 *fill[] 변수에, 그 화일의 크기는 sizes[]에 저장한 후 sortsz() 함수를 통해 그 크기가 큰 것부터 배열한다. 그리고 copy() 함수를 통해 실제 복사를 한다.

• sortsz(sizes, files, number)

일반 C에 준비되어 있는 qsort() 함수를 이용하지 않고 이 함수를 따로 만든 것은 구조체문을 사용하지 않고 여러 개의 화일을 소팅하기 위해서이다.

• myalloc(x)

이 함수는 malloc() 함수에 에러방지를 붙인 것이다. malloc은 memory allocation의 약자로 시스템에 메모리의 일부를 할당받아서 그 영역은 침범하지 못하게 배열을 만드는 것이다. 메모리 할당이 성공적이면 그 메모리의 번지를 돌려준다.

• copy(s, d, 1)

실질적으로 각 화일을 복사해주는 함수로서 읽기와 쓰기를 검사하고 에러가 발생하지 않으면 복사를 해 준다.

• addpath(dest, src1, src2)

이 함수는 dest pointer에 scr1과 scr2가 가리키는 문장을 합성해주는데 예제를 보면 쉽게 이해할 수 있을 것이다.

scr1에는 'a:\temp'와 scr2에는 'c:\action\tetris'가 있다면 dest에는 'a:\temp\ tetris'를 돌려준다.

• scdir(pat)

이 함수는 'pat'에 전달된 source의 path가 각각의 화일명으로 나누어 돌려주게 된다. 여기서 pat변수에 와일드카드('*')가 있는 경우 그에 해당하는 모든 화일들을 돌려주게 되는데 이 함수를 한번 부를때마다 한개의 화일명과 그 크기를 돌려준다. 만일 여러개의 화일이 존재한다면 이 함수를 여러번 불러주어야 할 것이다.

최초의 화일명은 도스 인터럽트 0X4E를 통해서 찾고 그 후에는 도스 인터럽트 0X4F를 통해 각각의 화일명을 받게 된다.

• freesp(drive)

이 함수는 목적디스켓의 여분을 알아내는 함수이다. 이것은 사용가능한 클러스터에 각 클러스터당 섹터수와 각 클러스터당 바이트 수를 곱하여 얻는다.

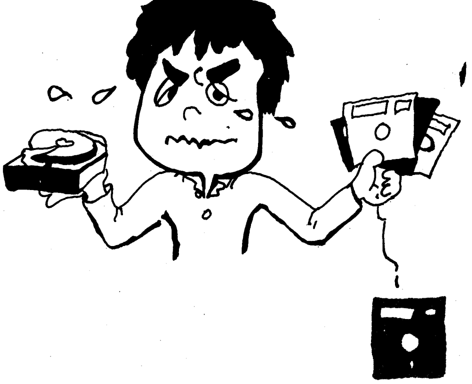
이 프로그램은 AZTEC C86-3.02e에서 작성된 것을 Turbo- C2.0으로 번역하여 실행시킨 것으로 컴파일시에는 옵션을 스몰모델*로 해주어야 한다.

이상으로 프로그램 설명은 마치기로 하고, 독자들에게 이 프로그램이 많은 도움이 되었으면 한다.

그러나 FILL도 만능은 아니다. FILL도 한번에 1024개 이상의 화일은 처리하지 못한다. 만약 1024개 이상의 화일을 선택하였다면 1024개 까지만 처리하고 그 이후의 화일은 무시해 버린다.

점차로 프로그램들의 크기가 커지고 기능이 복잡해져서 실제로 초보자나 전문적인 컴퓨터 유저가 아닌 입장에서 보면 부담만을 주는 현실에서 FILL과 같은 프로그램은 여간 반가운 일이 아닐 수 없다. FILL과 같이 누구나 쓰기 편리하고 간단한 프로그램들이 우리 유저들에게 더 많이 소개되었으면 한다. PC

*컴파일러 옵션의 모델은 프로그램을 컴파일하는데 사용할 메모리모델 선택에 사용한다. 디폴트는 small이며 이때는 프로그램 세그먼트가 최대 64KB, 데이터 세그먼트는 정적영역과 스택, 힙(heap)을 합하여 최대 64KB이다.



〔리스트 1〕 FILL.COM 소스 프로그램

1

```
#include <stdio.h>
#include <ctype.h>
#include <dos.h>
#define VERSION  "FILL - Dave Rand, Version 1.3  Sun Feb 15 13:25:37 1987"
-
/*
    Dave Rand
    72 Longfellow St.
    Thousand Oaks, CA  91360
    12/09/86

    AZTEC C86 - 3.20e

    This program may be freely distributed, in source and/or object,
    provided no fee is charged for it.

    FILL is a utility designed to allow you to fill floppies with multiple
    files. It will size the destination media automatically, and copy files
    to it in an efficient manner. New destination media is requested when the
    current media is full. If any files are too large to fit on the
    destination media, a message detailing this will appear on the screen.

    An invocation of fill would be as follows:

    fill [-q] source-path [source-path] destination-path

    Wildcard specifications are supported, as are paths.

    -q specifies quiet mode. In this mode, destination files are overwritten
    automatically if they exist. Normally, you are prompted if a file exists.

*/

#define MAXF 1024        /* 한번에 복사할 수 있는 화일의 최대수 */
#define SRCHFIR 0x4e
#define SRCHNXT 0x4f

long Fsize;              /* 현재 화일의 크기 */
int Curdrive;            /* 현재 사용중인 드라이브명 */
int Quiet=0;             /* 디폴트는  noisy로 */
```

2

```
main(argc,argv)          /* argc는 인자의 갯수이고, argv는 인자의 포
int argc;                   인터이다. */
char **argv;
{
    int c;
    union REGS r;

    if (argc < 3 ) {
        printf("%s\nUsage:\nfill [-q] source-path [source-path] dest-path\n",
        VERSION);
        exit(1);
    }


    r.h.ah=0x19;
    intdos(&r,&r);
    Curdrive = r.h.al;

    for (c = 1; c < argc-1; c++) {

      if (argv[c][0] == '-')
          switch(tolower(argv[c][1])) {
              case 'q':
                  Quiet = 1;
                  break;

              default:
                  printf("fill: Unknown option: %s\n",argv[c]);
                  break;
          }
    }


    for (c = 1; c < argc-1; c++)
        if (argv[c][0] != '-')
            do_copy(argv[c],argv[argc-1]);  /* 실제 복사할 함수를 부른
    r.h.ah=0x0e;                                다 */
    r.h.dl=Curdrive;
    intdos(&r,&r);
}
```

```
/*  do_copy(source,destination)
    do_copy finds and copies files from the source path to the destination,
    figuring out which files to copy first to ensure an optimum placement
    on the destination media.

*/

do_copy(source,dest)
char *source,*dest;
{
    char *files[MAXF],temp[100],*tc,*scdir();
    char *myalloc();
    int ff,x,t,user,count,curdrive;
    long sizes[MAXF],fsize(),freesp(),sprem;
    unsigned int leng;
    char source2[64];

    strcpy(source2,source);
    leng = strlen(source2);
    *(source2+leng)=0;
    if (dest[1] == ':')
        curdrive = toupper(dest[0])-'A'+1;
    else
        curdrive = Curdrive;

                                /* 우선, 복사할 드라이브를 지정 */
    for (ff=x=0; x<MAXF; x++) {
        if ((tc = scdir(source2))!=0) {
            files[x] = myalloc(strlen(tc) + 1);
            strcpy(files[x],tc);
            sizes[x] = Fsize;
            ff++;
        } else
            break;
    }


    if (ff ) {                  /* 화일이 발견될 경우 그 크기에 따라
        sortsz(sizes,files,ff)                  소팅시킨다. */
                                /* 데스티네이션에 복사된다 */
```

3

```
        for (count=0,x = ff; x > 0; ) {
        sprem = freesp(curdrive);
        for (t=ff-1; t>=0; t--) {
            if (sizes[t] != -1 && sprem > sizes[t]) {
                copy(files[t],dest,sprem);
                x--;
                count = 0;
                sizes[t] = -1;
                break;
            }
        }
        if (t < 0) {
            if (++count > 1) {
                printf("Could not mount sufficient space for:\n");
                for (t=0; t<ff; t++)    /*복사를 할 수 없는 경우 */
                    if (sizes[t] != -1)
                        printf("%14s %6ldK\n",files[t],
                                (sizes[t]+1023L)/1024L);
                exit(1);
            }
            for (sprem = 0L, t=0; t<ff; t++)   /* 앞으로 복사할
                                화일의 크기를 모두 구한다. */
                if (sizes[t] !=-1)
                if (sizes[t] !=-1)
                    sprem += sizes[t];

            printf("Please mount new disk on %s, %d files (%ldK) left to copy.\n
                    Press Enter to continue." ,dest,x,sprem/1024L);
            gets(temp);
        }
      }
    }
}

/*  sortsz(sizes, files, number)

    Sortsz sorts an array of longs into ascending sequence, and 'carries'
    the array files. Number is the total number of items to
    sort. sortsz uses a separate sort algorithm, since the built-in qsort
    does not have an external exchange facility, and thus cannot carry
```

4

```
    fields without using structures.
*/

sortsz(sizes,files,number)     /* 구조체문을 사용하지 않고 여러개의
char *files[];                    화일을 소팅하는 함수이다. */
long sizes[];
int number;
{
    register int i, j;
    int gap, k;
    long templ;
    char *temp2;

    for( gap=1; gap <= number; gap = 3*gap + 1 )
        ;

    for( gap /= 3;  gap > 0  ; gap /= 3 ) {
        for( i = gap; i < number; i++ ) {
            for( j = i-gap; j >= 0 ; j -= gap ) {
                if (sizes[j] > sizes[j + gap]) {
                    templ=sizes[j]; sizes[j]=sizes[j+gap]; sizes[j+gap]=templ;
                    temp2=files[j]; files[j]=files[j+gap]; files[j+gap]=temp2;
                }
            }
        }
    }
}


char  *myalloc(x)
/* Simply an error checking version of
   malloc()
*/
int x;
{
    char  *s;

    if ((s=(char  *)malloc(x))==NULL) {
        printf("Can't allocate %d bytes.\n",x);
        exit(1);
```

5

```
    }
    return(s);
}

/* copy(source, destination, bytes left)
    copy is the workhorse of the fill program, copying the source file to the
    destination. Note that a write error here is a fatal error, since the
    operating system just told us we had enough space for this file.
*/

copy(s,d,l)              /* 실질적으로 각 화일을 복사해주는 함수 */
char *s,*d;
long l;
{
    char  *buf;
    int fi,fo,len;
    char t[128],temp[128];

    addpath(t,d,s);
    buf=0;
    printf("%9ld bytes remaining. Copying %s to %s.\n",l,s,t);

    if ((fi=open(s,0))==-1) {
        printf("Can't open %s for input!\n",s);
        exit(2);
    }

    if (!Quiet) {                   /* Quite모드가 아닌 경우에 이미
                          존재하는 화일이 있는 경우 재복사를 묻는다 */
        if ((fo = open(t,0)) != -1) {
            close(fo);
            printf("fill: %s already exists, overwrite? (y/n) ",t);
            gets(temp);
            if (tolower(temp[0]) != 'y') {
                close(fi);
                exit(3);
            }
        }
    }
```

6

```
    if ((fo = creat(t,0644)) == -1) {
        printf("Can't create %s for write in copy!\n",t);
        exit(2);
    }
    close(fo);
    if ((fo = open(t,2)) == -1) {
        printf("Can't re-open %s for write!\n",t);
        exit(2);
    }
    buf = (char  *)myalloc(16384);

    len = _read(fi,buf,16384);
    while (len > 0) {
        if (_write(fo,buf,len) != len) {
            printf("Disk write error on file %s\n",t);
            exit(3);
        }
        len = _read(fi,buf,16384);
    }
    close(fi);close(fo);
    free(buf);
}

/* addpath(dest,src1,src2)

    addpath make a new filename from a destination path, plus a source
    path. For example, passed 'c:\temp' and 'd:/new/file/name/temp2',
    addpath will return in dest 'c:\temp\temp2'.
*/

addpath(dest,src1,src2)     /* 목적 포인터에 scr1과 scr2가 가리키는
char *dest,*src1,*src2;        문장을 합성 */
{
    register char *t;
    char *strchr();

    strcpy(dest,src1);
    dest += strlen(dest)-1;     /* 디렉토리의 마지막을 가리킨다. */
    if (strchr(":\\/",*dest++) == 0)    /* 끝에 '\'가 없으면 더한다. */
        *dest++ = '\\';
```

7

```
    t = src2 + strlen(src2) - 1;        /* 마지막을 가리킨다. */

    while ((t != src2) && (strchr(":\\/",*t) == 0))
        t--;                            /* 마지막 디렉토리를 가리킨다.*/
    if (t != src2)
        t++;                            /* 마지막 '\'를 가리킨다.*/

    strcpy(dest,t);                     /* 결과에 path를 더한다. */
}


/*  scdir(path)
    scdir returns the next matching file in the path, and returns the file's
    size in the global variable Fsize
*/

char *
scdir(pat)          /* pat에 전달된 소스 path가 각각의 화일명으로 나
char *pat;             누어 돌려주게 된다. */
{

    register char *cp, *np;
    char *strchr();

static int code = SRCHFIR;
static char nambuf[64], *tail;
static struct {
    char reserve[21];
    char attr;
    long mtime;
    long fsize;
    char name[13];
} srchbuf;
union REGS r;
struct SREGS s;
int te;

if (code == SRCHFIR) {
    if (strchr(pat, '*') == 0 && strchr(pat, '?') == 0) {
        setdta(&srchbuf.reserve[0]);
```

8

```
    r.h.ah=code;
    r.x.cx=0;                   /* 화일의 크기를 구한다. */
    r.x.dx=FP_OFF(pat);
    s.ds = FP_SEG(pat);
    intdosx(&r,&r,&s);
    if (r.x.ax)
       Fsize = 0;               /* 화일이 존재하지 않을 경우 크
    else                           기를 0으로 하고 돌려보낸다*/
       Fsize = srchbuf.fsize;
    code=0;
    return(pat);               /* 화일이름의 포인터를 보낸다.*/
  }

  strncpy(nambuf, pat, 64);
  for (cp = tail = nambuf ; cp < &nambuf[64] && *cp ; ++cp)
     if ((tail == nambuf && *cp == ':') || *cp == '/' || *cp == '\\')
        tail = cp+1;

}

setdta(&srchbuf.reserve[0]);      /* 처음 화일명이 아닌 이후의
                        화일명과 그 크기를 이 부분에서 구한다. */
r.h.ah=code;
if (code==SRCHFIR) {
   r.x.cx=0;
   r.x.dx=FP_OFF(pat);
   s.ds = FP_SEG(pat);
}
intdosx(&r,&r,&s);
te = r.x.ax;
if (code == 0 || te ) {
   code = SRCHFIR;
   return (char *)0;
}
```

9

```
code = SRCHNXT;

np = tail;
cp = srchbuf.name;
for (te=0;te<64;te++) {
   if ((*np = tolower(*cp))==0x0d || *np==0) {
      break;
   }
   np++;cp++;
}
Fsize = srchbuf.fsize;
   return(nambuf);
}


/*  freesp(drive)

    freesp determines the free space remaining on the specified (zero based)
    drive. The result is in bytes.
*/

long freesp(dr)          /* 목적 디스켓의 여분을 알아내는 함수 */
int dr;
{
    long Kspace;
    struct dfree s;

    getdfree(dr,&s);

    Kspace = (long)s.df_avail * (long)s.df_bsec * (long)s.df_sclus;

    return(Kspace);
}
```

10

프린터를 제어해주는 램상주 프로그램 PCNTL.COM

방재근 / 한양대 기계4 (HY-CORA)

프린터 사용시 폼피드나 라인피드 혹은 프린터를 껐다가 켜야 하는 일이 발생할 경우가 있다. 만약 프린터가 바로 자신의 컴퓨터 옆에 있다면, 자리에 앉은채로 이러한 동작을 할 수 있지만 그렇지 못하다면 바로 한뼘의 길이가 부족해도 자리를 일어나서 프린터를 재동작시켜야 하는 번거로운 일이 발생할 수 있다. 좀더 복잡한 곳이라면 짜증까지 낼 수 있겠지만, 여기서 소개하는 램상주 프로그램을 사용한다면 키보드만으로 이와같은 동작을 수행시킬 수 있을 것이다.

프린터사용시 불편한 점

필자가 처음 컴퓨터를 구입했을 때의 시스템 구성을 보면 메인메모리 640KB에 플로피디스크 2대, 허큘리스 비디오보드를 갖춘 IBM PC / XT 호환기종이었다. 이러한 구성의 시스템을 사용하는데 얼마동안은 그렇게 부족한 것을 느끼지 못했으나 시간이 갈수록 어떠한 한계를 느끼기 시작하였다. 그것은 프로그램이 대형화되어 적지 않은 응용 소프트웨어들이 많은 메모리를 차지하게 되었기 때문이며 이로인해 대용량의 하드디스크도 필요로 하게 되었다.

더우기 프로그래머에게 있어서 자기의 프로그램의 소스리스트를 프린터로 인쇄해서 볼 수 있다는 것은 좀더 많은 프로그램의 개발과 능률적인 결과를 가져올 수 있다고 생각한다. 이에 필자는 금년 1학기말 쯤에 주변기기로 하드디스크와 프린터를 구입하게 되었는데, 이중 프린터를 이용하여 필자 자신의 프로그램이나 공개 소프트웨어 등의 문서 화일 등을 프린트하면서 느낀 불편한 점 몇가지를 소개하고 이의 방법을 제시해 보고자 한다.

프린터 사용에 있어서 초기화나 라인피드, 폼피드 등을 하기 위해서는 몇번이고 자리를 이동하여 프린터에 있는 전원스위치나 라인피드, 폼피드 등의 스위치를 직접 조작해야만 하는 일이 빈번히 발생하여 처음에는 DOS 유틸리티 중의 하나인 디버그를 이용한 간단한 프로그램을 만들어 사용했었다(화면 1, 2, 3 참조).

이 세가지 리스트의 입력방법은 디버그를 로드한 후 각각의 리스트에 나타난 순서대로 입력한다. 여기서 처음의 A는 1행 어셈블러라하여 디버그상에서 소스코드를 입력할 수 있는 명령어이며 종료시는 리턴키를 누르면 된다. 100이라는 숫자는 100H번지부터 입력한다는 뜻으로 왜 100H번지부터 입력하는 지는 이후에 설명하도록 하겠다.

R은 현재 레지스터 값을 화면에 출력시키면서 값을 변경시킬 수 있는 명령어로 여기서 레지스터 CX의 값 변경은 화일의 길이를 기록하기 위함이고, N은 입력한 리스트의 화일이름을 지정하는 명령이다.

끝으로 W 명령은 위에서 입력한 리스트를 지정한 화일이름으로 디스크에 기록하는 명령어이다. 입력시 주의할 점은 디버그상의 숫자는 모두 16진수이므로 숫자 뒤에 H를 첨가해서 16진수라는 것을 표시할 필요가 없다는 것이다.(첨가시 에러 메시지가 나와 재입력을 요구한다.)

각각의 리스트에 대해서 이러한 과정을 마치고 나면 IP.COM, LF.COM, FF.COM이라는 세가지 실행 화일이 생성되는데 IP.COM은 프린터를 초기화(프린터 전원을 켰을때

〔화면 1〕 IP.COM 화일

```
C:\PC>DEBUG
-A100
32EA:0100 MOV  AH,01
32EA:0102 MOV  DX,00
32EA:0105 INT  17
32EA:0107 INT  20
32EA:0109
-RCX
CX 0000
:9
-NIP.COM
-W
Writing 0009 bytes
-Q

C:\PC>_
```

〔화면 2〕 FF.COM 화일

```
C:\PC>DEBUG
-A100
32EA:0100 MOV   AH,00
32EA:0102 MOV   AL,0C
32EA:0104 MOV   DX,00
32EA:0107 INT   17
32EA:0109 INT   20
32EA:010B
-RCX
CX 0000
:B
-NFF.COM
-W
Writing 000B bytes
-Q

C:\PC_
```

〔화면 3〕 LF.COM 화일

```
C:\PC>BEBUG
-A100
32EA:0100 MOV   AH,00
32EA:0102 MOV   AL,0A
32EA:0104 MOV   DX,00
32EA:0107 INT   17
32EA:0109 INT   20
32EA:010B
-RCX
CX 0000
:B
-NLF.COM
-W
Writing 000B bytes
-Q

C:\PC>_
```

의 상태)하는 화일이고, LF.COM은 라인피드(용지를 1라인 올림)의 역할을, FF.COM은 폼피드(용지를 1페이지 올림)의 역할을 수행하는 화일이다.

이 화일들의 실행은 프린터를 사용하기 전에 필요로하는 화일명을 입력하면 먼저 프린터가 입력된 화일 내용의 코드를 프린터로 출력시켜 마치 프린터를 직접 조작한 것과 같은 동작을 수행한다. 여기서 기술하는 내용은 이러한 역할의 화일을 하나로 묶어서 코드변환까지 해주고, 또 램상주프로그램으로 키보드상에서 제어가 가능하도록 작성된 프로그램을 소개하고자 하는 것이다.

COM화일과 EXE화일

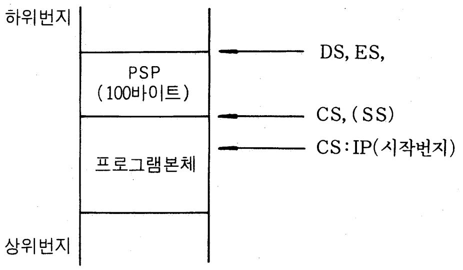
이 두 화일의 차이를 독자들은 벌써 잘 알고 있겠지만, 다시한번 여기서 간단히 정리를 하자면 DOS에서 실행될 수 있는 화일의 확장명은 모두 세가지로 COM, EXE, BAT가 있다. 이 중에서 기계어 코드로 되어 있는 것이 바로 확장명 COM과 EXE이라는 화일이다. 차이점은 COM화일을 로드하였을 때 CS, DS, ES, SS 등의 4개가 모두 일치하여 PSP의 시작을 지시하도록 세트한다.

여기서 PSP(Program Segment Prefix)란 사용자 프로그램영역 직전에 놓여 있는 부분으로서 프로그램이 실행하는데 필요한 여러가지 정보를 담고 있으며, 길이는 100H바이트이다. 또한 COM모델은 단일 세그먼트 구조로 64K내에 코드, 데이터, 스택, 그리고 엑스트라 등의 세그먼트가 모두 함께 있어야 한다. 따라서 프로그램의 크기가 64K를 넘을 수 없으며 실제로 PSP의 길이 256바이트와 강제적인 스택 길이 2바이트를 뺀 값이 화일의 최대 크기이다.

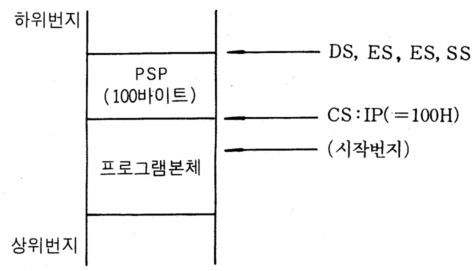
반면에 EXE모델은 프로그램이 메모리상에 로드되었을 때 DS와 ES는 PSP의 시작을, CS는 프로그램의 시작을 나타내도록 세트된다. 레지스터 SS를 특별히 지정하지 않으면 CS 레지스터와 같은 값이 된다.

그림 1은 COM모델과 EXE 모델에서 각각의 세그먼트

〔그림 1〕 EXE와 COM모델의 비교



(a) EXE모델의 세그먼트 초기치



(b) COM 모델의 세그먼트 초기치

초기치를 표시한 것인데 여기서 독자들은 100H라는 수치가 프로그램 실행시 필요한 정보를 포함하고 있는 영역임을 알 수 있을 것이다.

사용된 인터럽트

다음은 여기서 사용한 비디오출력, 키보드 입력, 프린터 출력, 램상주 인터럽트에 대해서 살펴보기로 하겠다.

• 인터럽트 10H

INT 10H는 비디오 출력에 관계되는 인터럽트로 이 프로그램에서는 화면출력에 관계되는 모든 사항에 사용하였다. 그 이유는 비디오 램에 직접 쓰는 것 보다 속도는 떨어지지만 비디오 카드의 호환성을 위한 것이기도 하다. 즉 허큘리스 카드의 비디오램은 세그먼트가 B000H부터지만 CGA카드는 B800H부터이다. INT 10H는 많은 루틴을 제공하고 있는데 여기서는 필자가 이용한 것만을 간략히 정리해 보기로 하겠다.

① **INT 10H/01H : 커서 형태 설정**

입력 : AH=01H
CH=커서의 시작 선
CL=커서의 끝 선
출력 : 새로운 커서 모양

(참고로 허큘리스카드의 디폴트는 시작이 0BH, 끝이 0CH이다. 만약 시작을 0EH, 끝을 0EH로 주면 화면상에서 커서가 사라지게 된다)

② **INT 10H/02H : 커서 위치 설정**

입력 : AH=02H
DH, DL=행, 열
BH=페이지 번호
출력 : 커서 위치가 변경

(좌측 상단의 시작은 행, 열 모두 0이다)

③ **INT 10H/03H : 현재 커서 위치를 얻음**

입력 : AH=03H
BH=페이지 번호
출력 : CH, CL=커서의 시작선과 끝선
DH, DL=커서의 행(y좌표)과 열(x좌표)

④ **INT 10H/08H : 현재 커서 위치에 있는 문자와 속성을 읽음**

입력 : AH=08H
BH=페이지 번호
출력 : AL=문자의 아스키 값
AH=문자의 속성 값

⑤ **INT 10H/09H : 현재 커서 위치에 문자와 속성을 씀**

입력 : AH=09H
AL=문자의 아스키 값
BH=페이지 번호
BL=속성 값
CX=출력될 문자 카운터
출력 : 지정한 위치에 문자와 속성이 출력

• 인터럽트 16H

INT 16H는 키보드 인터럽트로 세가지 서비스를 제공한다.

첫째는 키입력으로 레지스터 AH에 서비스번호 00H을 주고 인터럽트 16H을 부르면 레지스터 AH에는 스캔코드, AL에는 아스키값이 리턴된다. 여기서 스캔코드란 아스키값하고는 달리 키보드상의 키배열에 의한 값이다. 소스 리스트중 MAIN 프로시듀어에 있는 CMP AX, 00H에서 앞의 두바이트는 스캔 값, 뒤의 두바이트는 아스키값을 나타낸다.

두번째 서비스는 키가 눌려졌는지를 검사하고, 마지막으로 키보드 상태, 즉 CAPS LOCK, NUM LOCK, SCROLL LOCK 등의 상태를 검사하는데 좀더 자세한 사항은 인터럽트에 관한 매뉴얼을 참고해 주기 바란다.

• 인터럽트 17H

INT 17H은 프린터에 관한 인터럽트로 세가지 서비스를 제공한다. 첫째는 문자를 출력시키는 것이고, 다음은 프린터 초기화, 마지막은 프린터 상태를 검사하는 것이다.

다음은 각각에 대한 내용을 정리한 것이다.

① **INT 17H/00H : 레지스터 AL의 내용(문자)을 프린터로 출력**

입력 : AH=00H
AL=문자의 아스키 값
DX=프린터 번호(0, 1, 2)
출력 : AH=1→프린터 시간초과

② **INT 17H/01H : 프린터 초기화** (그림 2 참조).

입력 : AH=01H
DX=프린터번호(0, 1, 2)
출력 : AH=프린터상태

③ **INT 17H/02H : 프린터 상태를 읽음** (그림 2 참조).

입력 : AH=02H
DX=프린터 번호(0, 1, 2)
출력 : AH=프린터 상태

〔그림 2〕 프린터 상태 비트

```
프린터 상태(AH)
76543210
x.......  프린터 비사용중 (0=사용중)
.x......  프린터로부터 인지
..x.....  종이 없음
...x....  프린터 선택
....x...  I/O 에러
.....00.  사용 안됨
.......x  시간 초과 에러
```

•인터럽트 27H

INT 27H는 일명 Terminate and Stay Resident 인터럽트로 프로그램을 메모리에 상주시킨다. 이 인터럽트를 사용하기 위해서는 DS : DX에 프로그램의 마지막 부분에 있는 오프셋값을 주면 된다. 본 프로그램에서는 맨 마지막 프로시듀어의 오프셋값이 된다. 단점은 DOS 펑션 콜을 쓸 수 없으며 COM모델만을 지원하므로 64K이상의 프로그램은 램상주프로그램으로 이용할 수 없다는 것이다.

각 프로시듀어 설명

변수 NOWBAR는 현재 인버스 된 곳의 번지를 기억하고 있으므로 Up Key와 Down Key가 눌러질 때마다 가감된다.

MAIN PROC

ALT+F2키가 눌러졌는지를 검사하고, 맞으면 MENU_ON이라는 루틴을 부르고, 인터럽트 16H를 통하여 키입력을 받아 각각의 처리루틴을 부른다. END키가 눌러지면 MENU_OFF라는 루틴을 부른 후 처음 커서의 위치와 모양으로 복원시키고 제어를 DOS로 돌린다.

INI_PRINTER PROC

프린터를 초기화한다. 즉 인터럽트 17H의 서비스 1에 해당된다.

PRN_STATUS PROC

현재 프린터 상태를 얻는다. 인터럽트 17H의 서비스 2를 실행시키면서 AH에 상태가 리턴된다. 여기서 4번째 비트를 비트 조작을 하여 에러가 있으면 에러 플래그를 세트하고 (STC) BEEP를 호출한다.

BEEP PROC

퍼스널 컴퓨터 4월호 시스템 분석/IBM PC의 이해(4) 사운드를 참고했다.

UP_KEY PROC

현재 속성이 인버스인 부분을 정상으로 하고 바로 윗 부분을 인버스시킨다.

DOWN_KEY PROC

UP_KEY와 반대의 역할을 한다.

RIGHT_KEY PROC

프린터 상태를 점검하여 이상이 없으면 폼피드 시킨다.

LEFT_KEY PROC

프린터 상태를 점검하여 이상이 없으면 라인피드시킨다.

PGUP_KEY PROC

라인피드를 5회 시킨다.

RETURN_KEY PROC

현재 인버스된 곳의 코드를 프린터에 보낸다. 코드가 정의된 변수는 MENU_CODE_ON으로 코드가 FFH일 때까지 보낸다.

READ PROC

화면 왼쪽 상단의 문자와 속성을 읽어 STORAGE라는 변수에 저장한다. 읽는 방식은 인터럽트 10H의 서비스 8을 이용하는데 이 방법의 장점은 따로 비디오 카드의 종류를 검사할 필요가 없고 단점은 비디오 램에 직접 쓰는 것 보다 속도가 느리다는 것이다.

MENU_ON PROC

이 루틴은 일종의 초기화과정으로써 현재 커서의 위치와 모양을 변수에 저장하고 메뉴를 그린다.

PRINT PROC

실질적으로 원래의 화면 왼쪽 상단 부분을 1행씩 쓴다.

MENU_OFF PROC

PRINT 루틴을 불러 화면을 원래대로 복원시킨다.

STCURSOR PROC

지정한 위치로 커서를 이동시킨다.

WRITE PROC

UP_KEY와 DOWN_KEY 루틴의 서브루틴으로 속성에 맞춰 메뉴를 화면에 출력시킨다.

INSTALL PROC

인터럽트 벡터 테이블을 변경하고 인터럽트 27H를 이용하여 램상주 시킨다.

코드 변환 방법

코드는 MENU_CODE_ON에 정의되어 있는데, 각 명령당 7바이트로 구성되어 있고, FFH는 끝을 의미한다. 따라서 자신의 프린터에 알맞게 하기 위해서는 프린터에 맞는 코드로 변경한 후 FFH로 끝임을 알려주고 어셈블과 링크를 다시 하든지 아니면 PCTOOLS이나 Norton Utilities의 View/Edit 기능을 이용하여 PCNTL.COM을 직접 수정하는 것도 가능하다(**화면 4** 참조).

실행화일 생성방법

소스 리스트는 에디터를 사용하여 입력시키고, 어셈블과 링크를 하여 PCNTL.COM을 생성함으로써 실행화일을 얻는다. 리스트에서 ;는 주석문을 표시하는 것으로 입력시 제외시켜도 프로그램 실행에는 지장이 없다. 위에서 작성한 리스트를 어셈블하고 링크시키는 방법으로 볼랜드사의 터보 어셈블러와 마이크로 소프트사의 매크로 어셈블러 소프트웨어가 있는데 여기서 필자는 볼랜드사의 터보 어셈블러를 이용하였고, 매크로 어셈블러 사용자를 위해 **화면 5**에서 두가지 방법을 같이 보여주었다.

여기서 터보 어셈블러의 마지막 링크 과정시 /t라는 옵션을 주었는데 이 옵션의 의미는 실행화일인 COM화일을 직접 생성하라는 것이다. 만약 매크로 어셈블러를 사용한다고 하면 링크과정 후에 생성된 EXE화일은 EXE2BIN화일을 이용해서 램상주 프로그램으로 실행할 수 있는 COM화일

로 만들어 주어야 한다.

이 두 어셈블러의 차이를 어느 잡지에서 객관적으로 비교 설명한 적이 있었는데 필자의 경우는 디버그에 의해 프로그램 추적시 터보 디버그가 코드뷰보다 좀 더 정확하고 안정성이 있어 터보 어셈블러를 사용하게 되었다.

사용방법

위의 과정으로 PCNTL.COM 화일이 생성되었음을 확인하였으면 PCNTL이라고 입력한다. 그러면 잠시후 본 프로그램이 램에 상주하게 되는데 메뉴를 보기위해서는 Alt-F2키를 누른다. 메뉴에 있는 각각의 명령은 프린터 코드를 선택하는 것으로 선택된 명령의 코드를 프린터로 출력시켜 마치 프린터를 직접 제어하는 것과 같은 효과를 얻게 된다. 만약 선택된 명령을 취소하기 위해서는 마지막에 있는 리세트를 선택하여 지금까지 선택되었던 명령을 모두 취소시키면 된다.

〔화면 4〕 PC Tools에 의한 제어코드 확인

```
PC Tools Deluxe R4.11                                   Vol Label=WONGO
------------------------------File View/Edit Service----------------------------
Path=A:\BANG\*.*
File=PCNTL.COM     Relative sector 00000, Clust 00088, Disk Abs Sec 00184

Displacement ---------------- Hex codes------------------      ASCII value
 0000(0000)  E9 9F 05 00 2B 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D      +----------
 0016(0010)  2D 2D 2D 2D 2B 01 7C 20 43 6F 6E 64 65 6E 73 65   ----+ ¦ Condense
 0032(0020)  64 20 20 20 20 20 7C 02 7C 20 44 6F 75 62 6C 65   d     ¦ ¦ Double
 0048(0030)  2D 73 74 72 69 6B 65 20 7C 03 7C 20 44 6F 75 62   -strike ¦ ¦ Doub
 0064(0040)  6C 65 2D 77 69 64 65 20 20 20 7C 04 7C 20 44 72   le-wide   ¦ ¦ Dr
 0080(0050)  61 66 74 20 20 20 20 20 20 20 20 20 7C 05 7C 20   aft         ¦ ¦
 0096(0060)  45 6C 69 74 65 20 20 20 20 20 20 20 20 20 7C 06   Elite         ¦
 0112(0070)  7C 20 45 6D 70 68 61 73 69 7A 65 64 20 20 20 20   ¦ Emphasized
 0128(0080)  7C 07 7C 20 49 74 61 6C 69 63 73 20 20 20 20 20   ¦ ¦ Italics
 0144(0090)  20 20 7C 08 7C 20 50 69 63 61 20 20 20 20 20 20     ¦.¦ Pica
 0160(00A0)  20 20 20 20 7C 09 7C 20 52 6F 6D 61 6E 20 20 20       ¦ ¦ Roman
 0176(00B0)  20 20 20 20 20 20 7C 0A 7C 20 53 61 6E 73 20 53         ¦ ¦ Sans S
 0192(00C0)  65 72 69 66 20 20 20 20 7C 0B 7C 20 55 6E 64 65   erif      ¦ ¦ Unde
 0208(00D0)  72 6C 69 6E 69 6E 67 20 20 20 7C 0C 7C 20 52 45   rlining   ¦ ¦ RE
 0224(00E0)  53 45 54 20 20 20 52 45 53 45 54 20 7C 0D 2B 2D   SET   RESET ¦ +-
 0240(00F0)  2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2B 0F   -------------+
 0256(0100)  FF 00 00 00 00 00 1B 47 FF 00 00 00 00 0E FF 00         G
 0272(0110)  00 00 00 00 1B 78 00 FF 00 00 00 1B 4D FF 00 00      x     M
 0288(0120)  00 00 1B 45 FF 00 00 00 00 1B 34 FF 00 00 00 00     E     4
 0304(0130)  1B 50 FF 00 00 00 00 1B 78 01 1B 6B 00 FF 1B 78    P     x k   x
 0320(0140)  01 1B 6B 01 FF 1B 2D 01 FF 00 00 00 1B 40 FF 00     k  -     @
 0336(0150)  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Home=beg of file/disk  End=end of file/disk
ESC=Exit  PgDn=forward  PgUp=back  F1=toggle mode  F2=chg sector num  F3=edit
```

다음에 이 프로그램에서 사용된 키와 기능을 요약했다.

ALT+F2—화면 왼쪽 상단에 메뉴 화면이 나타난다(화면 6).
HOME—프린터를 초기화한다. 처음 전원을 ON한 것과 같다.
UP—메뉴를 상단으로 이동시킨다.
PGUP—라인피드를 5회 반복한다.
LEFT—폼 피드
RIGHT—라인 피드
END—프로그램을 끝내고 제어를 DOS로 복귀시킨다.
DOWN—메뉴를 하단으로 이동시킨다.
RETURN—메뉴를 선택한다.

개선되어야할 점

첫째로 메모리상에서 프로그램이 필요치 않을 경우 램상

〔화면 5〕 터보와 매크로 어셈블러 실행과정

```
C:\PC>TASM PCNTL
Turbo Assembler  Version 1.0  Copyright (c) 1988 by Borland International

Assembling file:   PCNTL.ASM
Error messages:    None
Warring messages:  None
Remaining memory:  317k

C:\PC>TLINK/t PCNTL
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

C:\PC>_
```

(a) 터보 어셈블러인 경우

```
C:PC>MASM PCNTL;
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All right reserved.
  49676 + 262656 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors

C:\PC>LINK PCNTL;

Microsoft (R) Overlay Linker  Version 3.65
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

LINK : warring L4021: no stack segment

C:\PC>EXE2BIN PCNTL.EXE PCNTL.COM

C:\PC>_
```

(b) 매크로 어셈블러인 경우

주 프로그램을 제거할 수 있는 프로그램이 빠져 있다는 것이다. 이것은 자료가 미비한 관계로 시도를 못했는데 자료가 준비되면 다시 한번 소개할 수 있는 기회가 주어졌으면 한다.

둘째는 필자의 프린터가 LX-800인데 모든 기능을 여기에 맞추다 보니 다른 프린터와의 호환 및 한글 처리가 상당히 미비하다. 영문 모드인 경우는 별 무리가 없지만 한글인 경우는 본 프로그램의 몇몇 명령어가 작용하지 않아 큰 아쉬움이 남는다. 또한 낱장 공급 장치가 있는 경우에는 폼 피드가 작용하지 않았으며, 프린터 버퍼가 있어 여러대의 컴퓨터가 한대의 프린터를 이용할 때는 프린터 초기화가 작용하지 않았다. 실력있는 독자의 참여를 기대해 본다.

〔화면 6〕 램상주된 프로그램의 메뉴

```
+--------------+
¦ Condensed    ¦
¦ Double-strike¦
¦ Double-wide  ¦A is WONGO
¦ Draft        ¦\BANG
¦ Elite        ¦
¦ Emphasized   ¦>       1-01-80   3:31a
¦ Italics      ¦>       1-01-80   3:31a
¦ Pica         ¦2560    7-10-89  10:18a
¦ Roman        ¦1476    1-01-80   3:23a
¦ Sans Serif   ¦>       7-13-89   6:03a
¦ Underlining  ¦  12    1-01-80   3:36a
¦ RESET  RESET ¦5232    7-09-89   1:09p
+--------------+2244    1-01-80  12:14a
IP       COM       9    7-05-89   2:32a
FF       COM      11    7-05-89   2:33a
LF       COM      11    7-05-89   2:34a
PCNTL          13440    7-09-89  11:56a
        12 File(s)     77824 bytes free

A:\BANG>
```

맺음말

실제 프로그램의 설명보다는 관련된 자료에 많은 지면을 할애했는데 아마도 실력있는 독자들은 좀 따분했을 것이다. 그러나 필자의 주위을 보면 언어는 알아도 제대로 프로그램을 짜는 사람이 드물어 나름대로 이유를 분석한 결과 컴퓨터의 내부처리 즉, 메모리 구성이나 인터럽트에 대해서 정확하고도 상세히 알고 있지 않다는 결론에 이르게 되었다. 한가지 예로서 파스칼은 알아도 터보 파스칼은 몰라 어느 한계 이상을 넘지 못하는 경우를 종종 보게 되었다. 이에 이미 많은 발표가 있어 굳이 언급할 필요가 없는 사항도 다시 한번 소개와 정리를 하였는데 깊은 이해가 있길 바란다.

〔리스트〕 PCNTL 프로그램의 소스리스트

```
INTERRUPTS      SEGMENT AT 0H
     ORG        9H*4                          ; 키보드 인터럽트
KEYBOARD_INT    LABEL   WORD
INTERRUPTS      ENDS
;
ROM_BIOS_DATA   SEGMENT AT 40H                ; BIOS 데이터 영역
        ORG     1AH
        HEAD    DW ?                          ;
        TAIL    DW ?
        BUFFER  DW 16 DUP (?)                 ; 키보드 버퍼
    BUFFER_END  LABEL WORD
ROM_BIOS_DATA   ENDS
;
CODE_SEG        SEGMENT
        ASSUME  CS:CODE_SEG
        ORG     100H                          ; ORG=100H (.COM화일지정)
FIRST:  JMP     INSTALL                       ;
;
NORMAL        EQU     07H                     ; 속성 정의
INVERSE       EQU     70H
MENU          DB      0,'+--------------+'
              DB      1,'¦ Condensed    ¦'
              DB      2,'¦ Double-strike¦'
              DB      3,'¦ Double-wide  ¦'
              DB      4,'¦ Draft        ¦'
              DB      5,'¦ Elite        ¦'
              DB      6,'¦ Emphasized   ¦'
              DB      7,'¦ Italics      ¦'
              DB      8,'¦ Pica         ¦'
              DB      9,'¦ Roman        ¦'
              DB     10,'¦ Sans Serif   ¦'
              DB     11,'¦ Underlining  ¦'
              DB     12,'¦ RESET  RESET ¦'
              DB     13,'+--------------+'
MENU_CODE_ON  DB     0FH,0FFH,5 DUP (0)            ; 축소 모드
              DB     1BH,47H,0FFH,4 DUP(0)         ; 이중 인쇄
              DB     0EH,0FFH,5 DUP(0)             ; 두배 폭 인쇄
              DB     1BH,78H,00H,0FFH,3 DUP(0)     ; DRAFT ON
              DB     1BH,4DH,0FFH,4 DUP (0)        ; ELITE ON
              DB     1BH,45H,0FFH,4 DUP (0)        ; 강조 모드
              DB     1BH,34H,0FFH,4 DUP (0)        ; 이텔릭체
              DB     1BH,50H,0FFH,4 DUP (0)        ; PICA ON
```

```
                DB      1BH,78H,01H,1BH,6BH,00H,0FFH  ; 로마체
                DB      1BH,78H,01H,1BH,6BH,01H,0FFH  ; SANS SERIF ON
                DB      1BH,2DH,01H,0FFH,3 DUP (0)    ; 밑줄문자
                DB      1BH,40H,0FFH, 4 DUP (0)       ; RESET
OLD_KEYBOARD    DW      2 DUP(?)
STORAGE         DB      17*14*2 DUP (?)               ; 스크린 버퍼
NOWBAR          DW      ?                             ; 메뉴의 현재 오프세트값
CURSOR_R        DB      ?                             ; 커서 행 위치
CURSOR_C        DB      ?                             ; 커서 열 위치
CURSOR_S        DB      ?                             ; 커서 시작 라인
CURSOR_E        DB      ?                             ; 커서 끝 라인
;
MAIN            PROC    NEAR
                ASSUME  CS:CODE_SEG
                STI                                   ; 인터럽트 가능
                PUSH    AX                            ; 레지스터를 스택에 저장
                PUSH    BX
                PUSH    CX
                PUSH    DX
                PUSH    DI
                PUSH    SI
                PUSH    DS
                PUSH    ES
                PUSHF
                CALL    DWORD PTR OLD_KEYBOARD        ; INT 9H 호출
                ASSUME  DS:ROM_BIOS_DATA
                MOV     BX,ROM_BIOS_DATA
                MOV     DS,BX
                MOV     BX,TAIL
                CMP     BX,HEAD
                JNE     DO_PROGRAM                    ; 버퍼가 비었으면 EXIT
                JMP     FINISH
DO_PROGRAM:     SUB     BX,2
                CMP     BX,OFFSET BUFFER
                JAE     NO_WRAP
                MOV     BX,OFFSET BUFFER_END
                SUB     BX,2
NO_WRAP:        MOV     DX,[BX]
;
                CMP     DX,6900H                      ; Alt + F2 키입력 비교
                JNE     FINISH
                MOV     TAIL,BX                       ; 키스트로크 Disable
;
                PUSH    CS                            ; CS의 내용을 DS로
                POP     DS
;
                CALL    MENU_ON                       ; 메뉴 디스플레이
;
REPEAT_UNTIL:   MOV     AH,00H                        ; 입력된 키 읽음
                INT     16H
;
                CMP     AX,4F00H                      ; END 키 인지 비교
                JE      FIN                           ; 맞으면 EXIT
                CMP     AX,4700H                      ; HOME키=프린터초기화
                JNE     IS_UP_KEY
                CALL    INI_PRINTER
                JMP     REPEAT_UNTIL
IS_UP_KEY:      CMP     AX,4800H                      ; 위 화살표 키
                JNE     IS_DOWN_KEY
                CALL    UP_KEY
                JMP     REPEAT_UNTIL
IS_DOWN_KEY:    CMP     AX,5000H                      ; 아래 화살표 키
                JNE     IS_RETURN_KEY
                CALL    DOWN_KEY
                JMP     REPEAT_UNTIL
IS_RETURN_KEY:CMP       AX,1C0DH
                JNE     IS_LEFT_KEY
                CALL    RETURN_KEY
                JMP     REPEAT_UNTIL
IS_LEFT_KEY:    CMP     AX,4B00H
                JNE     IS_RIGHT_KEY
                CALL    LEFT_KEY
                JMP     REPEAT_UNTIL
IS_RIGHT_KEY:   CMP     AX,4D00H
                JNE     IS_PGUP_KEY
                CALL    RIGHT_KEY
                JMP     REPEAT_UNTIL
IS_PGUP_KEY:    CMP     AX,4900H
                JNE     REPEAT_UNTIL
                CALL    PGUP_KEY
                JMP     REPEAT_UNTIL
;
FIN:            CALL    MENU_OFF                      ; 스크린영역 재표시
;
                MOV     AH,01H                        ; 커서 모양 재표시
                MOV     CH,CURSOR_S                   ;
                MOV     CL,CURSOR_E
                INT     10H
;
                MOV     AH,02H                        ; 커서 위치 재표시
                MOV     BH,00H                        ;
                MOV     DH,CURSOR_R
                MOV     DL,CURSOR_C
                INT     10H
;
FINISH:         POP     ES
                POP     DS
                POP     SI
                POP     DI
                POP     DX
                POP     CX
                POP     BX
                POP     AX
                IRET
MAIN            ENDP
;
```

```
INI_PRINTER   PROC    NEAR                    ; 프린터 초기화
              MOV     AH,01H                  ; 서비스 = 1
              MOV     DX,00H
              INT     17H
              RET
INI_PRINTER   ENDP
;
PRN_STATUS    PROC    NEAR                    ; 프린터 상태 읽음
              MOV     AH,02H
              MOV     DX,00H
              INT     17H
              TEST    AH,8                    ; Bit 4 = I/O 에러
              JE      PRINTER_OK
              STC                             ; 에러 플래그 세트
              CALL    BEEP
              RET
PRINTER_OK:   CLC
              RET
PRN_STATUS    ENDP
;
BEEP          PROC    NEAR
              IN      AL,61H
              OR      AL,3
              OUT     61H,AL
              MOV     AL,10110110B
              OUT     43H,AL
              MOV     AX,2705
              OUT     42H,AL
              MOV     AL,AH
              OUT     42H,AL
              MOV     CX,1000H
DELAY:        LOOP    DELAY
              IN      AL,61H
              AND     AL,11111100B
              OUT     61H,AL
              RET
BEEP          ENDP
;
UP_KEY        PROC    NEAR
              MOV     BL,NORMAL               ; 보통속성을 가진
              MOV     SI,NOWBAR               ;  현재 문자 재기록
              CALL    WRITE                   ;
              MOV     BL,INVERSE
              CMP     BYTE PTR [SI],1         ; CURRENT=TOP이면
              JE      TOP                     ;  마지막 문자로 이동
              SUB     SI,12H
              JMP     END_OF_UP_KEY
TOP:          ADD     SI,12H*11
END_OF_UP_KEY:
              CALL    WRITE                   ; 반대속성을 가진 다음
              MOV     NOWBAR,SI               ;  다음 문자 재기록
              RET                             ;
UP_KEY        ENDP
;
DOWN_KEY      PROC    NEAR
              MOV     BL,NORMAL
              MOV     SI,NOWBAR
              CALL    WRITE
              MOV     BL,INVERSE
              CMP     BYTE PTR [SI],12
              JE      BOTTOM
              ADD     SI,12H
              JMP     END_OF_DOWN_KEY
BOTTOM:       SUB     SI,12H*11
END_OF_DOWN_KEY:
              CALL    WRITE
              MOV     NOWBAR,SI
              RET
DOWN_KEY      ENDP
;
RIGHT_KEY     PROC    NEAR
              CALL    PRN_STATUS              ; 프린터 체크
              JC      END_OF_RIGHT            ; If Error, Return.
              MOV     AH,00H                  ; Else FormFeed.
              MOV     AL,0AH                  ; 0AH = FormFeed
              MOV     DX,0
              INT     17H
END_OF_RIGHT: RET
RIGHT_KEY     ENDP
;
LEFT_KEY      PROC    NEAR
              CALL    PRN_STATUS              ; 프린터 체크
              JC      END_OF_LEFT             ; If Error, Return.
              MOV     AH,00H                  ; Else LineFeed.
              MOV     AL,0CH                  ; 0CH = LineFeed
              MOV     DX,0
              INT     17H
END_OF_LEFT:  RET
LEFT_KEY      ENDP
;
PGUP_KEY      PROC    NEAR
              CALL    PRN_STATUS              ; 프린터 체크
              JC      END_OF_PGUP             ; If Error, Return.
              MOV     CX,05H                  ; Else LineFeed 5 Times.
LF5:          MOV     AH,00H                  ; CX = Count
              MOV     AL,0AH
              MOV     DX,0
              INT     17H
              LOOP    LF5
END_OF_PGUP:  RET
PGUP_KEY      ENDP
;
RETURN_KEY    PROC    NEAR
              CALL    PRN_STATUS
              JC      RETURN_MAIN
              MOV     DI,OFFSET MENU_CODE_ON
```

```
            MOV     SI,NOWBAR
            MOV     AX,0                    ; AX를 초기화
            MOV     AL,BYTE PTR [SI]
            DEC     AL
            MOV     BX,0                    ; BX를 초기화
            MOV     BL,07H
            MUL     BL
            ADD     DI,AX
CONTINUE:   MOV     AH,00H                  ; 프린터로 코드를 보냄
            MOV     AL,BYTE PTR[DI]
            MOV     DX,0
            INT     17H
            INC     DI
            CMP     BYTE PTR[DI],0FFH       ; 코드가 FFH일때까지
            JNE     CONTINUE
RETURN_MAIN: RET
RETURN_KEY  ENDP
;
READ        PROC    NEAR
            PUSH    CX
            MOV     CX,17
NEXT:       CALL    STCURSOR
            MOV     AH,08H
            MOV     BH,00H
            INT     10H
            MOV     BYTE PTR [DI],AH
            MOV     BYTE PTR [DI+1],AL
            ADD     DI,2
            LOOP    NEXT
            POP     CX
            RET
READ        ENDP
;
MENU_ON     PROC    NEAR
            MOV     AH,03H                  ; 현재 커서의 위치와
            MOV     BH,00H                  ;  모양 저장
            INT     10H                     ; 인터럽트 10H 사용
            MOV     CURSOR_R,DH             ; INT 10H/ 03H
            MOV     CURSOR_C,DL             ; RETURN
            MOV     CURSOR_S,CH             ; DH=행,  DL=열
            MOV     CURSOR_E,CL             ; CH=시작,CL=끝
            MOV     AH,01H                  ; 커서 제거
            MOV     CX,0E0EH
            INT     10H
            MOV     DI,OFFSET STORAGE
            MOV     CX,14
            MOV     DX,0000H
DO:         CALL    READ
            MOV     DL,0
            INC     DH
            LOOP    DO
            MOV     NOWBAR,OFFSET MENU
            MOV     BL,NORMAL
            MOV     SI,NOWBAR
            CALL    WRITE
            MOV     BL,INVERSE
            ADD     NOWBAR,12H
            MOV     SI,NOWBAR
            MOV     CX,000DH
DO_WHILE:   PUSH    CX
            CALL    WRITE
            MOV     BL,NORMAL
            ADD     SI,12H
            POP     CX
            LOOP    DO_WHILE
            RET
MENU_ON     ENDP
;
PRINT       PROC    NEAR
            PUSH    CX
            MOV     CX,17
NEXTP:      CALL    STCURSOR
            MOV     AH,09H
            MOV     BH,00H
            MOV     BL,BYTE PTR [DI]
            MOV     AL,BYTE PTR [DI+1]
            INT     10H
            ADD     DI,2
            LOOP    NEXTP
            POP     CX
            RET
PRINT       ENDP
;
MENU_OFF    PROC    NEAR
            MOV     DI,OFFSET STORAGE
            MOV     CX,14
            MOV     DX,0000H
DOOFF:      CALL    PRINT
            MOV     DL,0
            INC     DH
            LOOP    DOOFF
            RET
MENU_OFF    ENDP
;
STCURSOR    PROC    NEAR
            MOV     AH,02H                  ; 커서 세트
            MOV     BH,00H                  ; 페이지 표시
            INT     10H
            INC     DL
            RET
STCURSOR    ENDP
;
WRITE       PROC    NEAR
            PUSH    SI                      ; 문자 시작 OFFSET 저장
            MOV     DL,0                    ; DL(열) 초기화
            MOV     DH,BYTE PTR [SI]        ; DH = 행
```

```
        MOV     CX,0012H
        DEC     CL
FOR_NEXT:       PUSH    CX
        CALL    STCURSOR
        INC     SI
        MOV     AH,09H                  ; 서비스 9H
        MOV     BH,00H                  ; 페이지 표시
        MOV     AL,[SI]                 ; 문자 기록
        MOV     CX,1                    ; 시간 기록
        INT     10H
        POP     CX
        LOOP    FOR_NEXT
        POP     SI                      ; SI 재기록
        RET
WRITE   ENDP
;
INSTALL PROC    NEAR
        ASSUME  DS:INTERRUPTS           ; 데이터 세스먼트가
        MOV     AX,INTERRUPTS           ;  인터럽트영역이 된다.
        MOV     DS,AX                   ; 구인터럽트루틴을 얻음
        MOV     AX,KEYBOARD_INT         ; OLD_KEYBOARD르 넣음
        MOV     OLD_KEYBOARD,AX
        MOV     AX,KEYBOARD_INT[2]
        MOV     OLD_KEYBOARD[2],AX
        MOV     KEYBOARD_INT,OFFSET MAIN        ; 키보드루틴의 번지 로드
        MOV     KEYBOARD_INT[2],CS
        MOV     DX,OFFSET INSTALL
        INT     27H
INSTALL ENDP
;
CODE_SEG        ENDS
        END             FIRST
```



WORDPROCESSOR

보석글에 날개를 단다
인쇄형식을 다양하게 하려면

박정래 / 인하대학교 컴퓨터써클(INCOM)

보석글로 신문을 간행한다? 굉장히 생소한 말이다. 신문을 만드는 일은 생각보다 어렵고 복잡하며 손이 많이 가는 작업이 아닐 수 없다. 왜냐하면 면의 특성에 따라 기사 등을 배치해야 하며 같은 면에서도 기사의 내용과 길이에 따라 재배치함은 물론이고 다양한 크기의 문자와 다양한 모양의 문자가 필요하기 때문이다. 또 상황에 따라 그림을 삽입해야 하며 시각적 효과를 높이기 위해 그래픽 문자나 라인 등을 사용해야 한다.

기존의 출판 프로그램인 DTP 등 여러 개의 출판전용 프로그램이 있기는 하지만 대부분의 프로그램이 사용자에게 생소하기 때문에 많이 쓰이지 않거나 아니면 다른 방법으로 신문을 간행하는 경우가 대부분이다.

예를 들어, 먼저 기사를 수집한 후에 내용을 정리 배치한다. 그후 거기에 맞게 기사별로 워드프로세서 등으로 작성한 후 프린트해서 스크랩하는 방법이 있다. 또한 먼저 기사를 수집 배치한 뒤, 작성할 때 직접 그 크기나 위치에 맞게 편집하는 방법이 있다.

위의 방법들은 굉장히 불편하고 손이 많이 간다. 이에 비해서 지금부터 소개하는 방법은 위의 방법들 보다 더 편리하고 비교적 조그만 신문, 잡지, 회지 등을 만들기에 적당하다.

보석글이 신문발간을 가능하게 하는 것은 보석글 II Ver 2.0에서 지원하는 복수컬럼 인쇄방식과 그래픽 인쇄방식이다.

여기서는 우선 복수컬럼 인쇄방식에 대해 설명하고자 하며 그래픽 인쇄는 다음 기회에 소개하고자 한다.

특히 복수컬럼 인쇄방식은 대단히 유용한 기능으로 사용자의 입장에서 신문, 잡지 뿐 아니라 보고서 등 여러 면에서 도움을 줄 것이다.

다양한 형태의 인쇄

보기 1에 있는 예제 화일을 보자. 대다수의 신문, 잡지에서 볼 수 있는 인쇄방식이다. 보통의 방법을 가지고 이같이 인쇄하고자 한다면 사용자는 보석글의 EDIT상태에서 라인과 컬럼을 확인해 가면서 일일이 인쇄하고자 하는 방식대로 타자해 넣어야 한다. 이러한 작업은 상당히 복잡하고 손이 많이 가

〔보기 1〕 복수컬럼 인쇄방식으로 출력한 예

4 써클 이모저모 Word Processor 1989年 6月 세째주

파란팀 힘겨운 '우승'

동문체육대회에서 발휘한 정신력

지난 6월 25일 본교 대운동장에서 있었던 동문체육대회에서 파란팀이 초록팀을 4-3 으로 제치며 힘겨운 우승을 차지했다.
이날 써클회원을 기수와 과의 구분 없이 무작위로 초록팀과 파란팀으로 나눈 뒤 순서에 따라 진행되었다.
써클 2기 선배님부터 막내 10기 회원까지 고루 참가하여 써클의 힘과 기량을 단합이라는 일체의 힘으로 표출한 가운데 3-3 으로 팽팽히 맞서던 중 마지막 경기인 축구를 파란팀의 일치된 힘을 바탕으로 박정래(9기)군이 2골을 기록하는 수훈에 힘입어 초록팀을 4-1 의 스코어로 이겼다.

▲ 종합전적

● 발야구
초록팀 2 - 1 파란팀
(1승) (1패)

● 닭싸움 (남)
초록팀 3 - 1 파란팀
(2승) (2패)

● 닭싸움 (여)
초록팀 1 - 2 파란팀
(2승1패) (1승2패)

● 피구
초록팀 0 - 3 파란팀
(2승2패) (2승2패)

초록팀 5 - 0 파란팀
(3승2패) (2승3패)

초록팀 0 - 2 파란팀
(3승3패) (3승3패)

● 축구
파란팀 ┌1 - 0┐ 초록팀
4 └3 - 1┘ 1
(4승3패) (3승4패)

►종합전적

파란팀 4 - 3 초록팀
(우승) (준우승)

♥♥♥♥♥♥♥♥ 게시판 ♥♥♥♥♥♥♥♥

°- 동문 체육 대회 뒤풀이

6월 25일 본교 대운동장에서 동문 체육 대회를 가진 뒤 본관 앞 잔디에서 1 차 뒤풀이를 가졌다.
각 기수별로 장기 자랑을 했으며 졸업한지가 오래되신 선배님을 위해 각 기수별로 자기소개도 했다.
다과와 함께 이루어진 이번 뒤풀이에서는 재미있는 소제로 행해진 장기자랑이 인기를 끌었으며 참석한 모든 회원을 웃음의 도가니로 이끌었다.

°- 인컴 정기총회 개최

본 써클 정기총회가 6월 28일 본교 학생회관 소강당에서 열린다.
이번 총회에서는 각부 한 학기 보고와 다음 학기 계획이 발표되고 또 회칙 개정안이 심의되며 이번 학기로 임기가 완료되는 현 회장의 뒤를 이어갈 새 회장이 선출된다.
회장 후보로는 이영우,김영호,황병선,이규영 후보가 추천을 받았으며 부회장 후보로는 정영숙,고영희 후보가 추천을 받았다.

는 지겨운 작업이다.

예를 들어 가운데 단에 새로운 라인을 삽입하고자 하는 경우 EDIT상태에서 작업을 하게 되면 전체를 재구성해야 하는 번거롭고도 힘든 작업을 해야 한다. 또 보기1과같이 확대 문자를 사용해야 하는 경우에는 중간 중간에 인쇄를 해서 컬럼을 확인하고 정렬해야 한다.

더우기 신문이나 잡지의 경우 대부분이 이러한 작업이기 때문에 위와 같은 방법으로는 불가능하다.

이제 이러한 작업을 손쉽게 할 수 있는 방법을 소개하고자 한다. 두가지 방법이 있는데 둘 다 보석글 II 버전 2.0에서 지원하는 인쇄형식 지정 명령을 사용하는 것이다.

(1) 컬럼 명령

형식 : .Column N1 N2 N3 …

한 페이지를 세로로 분할하여 보기 1과 같이 다른 내용을 한 라인에 존재하도록 인쇄해 주는 명령이다. 이 명령의 다음에 오는 화면상의 첫 페이지는 N1째 컬럼에서부터 인쇄되며 화면상의 그 다음 페이지는 같은 장의 N2째 컬럼에서부터 인쇄되고 그 다음 페이지는 N3째 컬럼에서부터 인쇄된다.

그러므로 EDIT상태에서 문장 폭을 적당히 조절하면 한 장에 여러 단으로 나누어 인쇄할 수 있다. 그러나 문장 폭을 적절히 조정하지 않으면 겹쳐서 인쇄되는 경우가 있으니 주의해야 한다.

화면 1과 같은 경우, 'I'를 4배 확대한 후 다음 라인에 '.new' 옵션을 주어 강제로 새 페이지를 지정했다.

'.new'전까지 첫 페이지로 간주되어 앞의 '.column 1 1 33 65' 옵션에 따라 1컬럼부터 인쇄되었으며 '.new' 다음부터 한 페이지의 분량(58라인)이 두번째 페이지로 간주되어 앞의 옵션에 따라 1컬럼부터, 다음 페이지의 분량이 33컬럼부터, 그 다음 페이지가 65컬럼부터 인쇄가 되었다.

참고로 '.new' 다음의 빈공간(4라인)은 두번째 페이지의 인쇄위치를 'I'와 같은 라인으로 하기 위함이다.

'.column N1 N2 N3 …'의 인쇄방식에 대해 알아 보았는데 이 방법은 신문이나 잡지, 회지 등의 복잡한 편집에는 적합하지 않다.

〔화면 1〕 컬럼명령의 입력

```
라인:    1( 총 179) 화일명: EXAMPLE0.GEM  모드: [ 매뉴얼 ]  쓴영역: 4( 총 291K)
[ 삽입중 ] [보통문자][ Ragged Align ]
<......+.......+.......+.....>.+.......+.......+.......+.......+.......+.......
.........1.........2.........3.........4.........5.........6.........7.........
1234567890123456789012345678901234567890123456789012345678901234567890123456789
.column 1 1 33 65
제 1 장  서  론

제 1 절 연구의 목적

I
.new




    llegitimacy 란 말은 혼외자
    를 뜻하는 말이다.
혼외자의 자위는 나라와  시대에
따라 다르나 서양에서는  일부일
처제의 혼인을 정당시하고  정당
한 혼인에 의하지 않는 성결합을
을 죄악으로 보고   죄있는 결합
의 죄없는 과실  인 혼외자를 차
별하기 시작하였다. 그러나 근대
```

왜냐하면 이 방식은 세밀하고 정교하지가 못하기 때문이다.

(2) 세이브와 리턴

형식 : .save x1
 ⋮
.return x1

인쇄위치를 상대적인 기준에 의해서 설정하는 인쇄 옵션으로서 x1는 a에서 z까지의 영문 자음만 대체될 수 있다. '.save'명령은 그 위치를 기억하게 하는 것으로 컬럼과 라인이 동시에 기억된다. '.return'명령은 인쇄위치를 '.save'에 의해서 기억된 위치로 이동하게 하는 것이다.

예를 들어 보석글로 인쇄명령을 주면 '.save x1'의 위치를 기억해 두었다가 '.return x1'명령을 만나면 인쇄위치를 'save x1'의 위치로 옮겨 프린트하게 된다.

'save'와 'return'명령을 사용하여 작성한 예제를 보면서 자세히 알아 보자.

화면 2에서 맨 처음의 'save a'는 처음의 위치를 보석글이 기억하게 한다. 다음의 'save b'에 의해서 또 하나의 위치가 기억된다. 보석글에 인쇄 명령을 주게 되면 처음('save a')부터 4배 확대의 'I'까지 인쇄된 후에 'return b'에 의해서 '.save b'의 위치('I'가 인쇄된 라인)로 이동해 'return b' 다음의 내용이 4배 확대의 'I'가 인쇄된 라인에 인쇄된다.

그리고 다음의 '.save c'와 '.return c'에 의해서 4배 확대의 '혼'과 'return c' 다음의 내용이 같은 라인으로부터 인쇄된다. 그런 뒤에는 그 다음에 나오는 'return a'와 '.indent 33'에 의해서 그 다음의 내용이 맨 첫줄의 33컬럼부터 인쇄된다(보기 2 참조).

그 다음에 나오는 'return a', 'indent 65', 'save d', 'return d'도 앞에 설명한 것과 동일하다.

다음에 나와 있는 결과를 보면 자세히 이해가 될 것이다.

〔화면 2〕 세이브와 리턴명령 입력

```
라인:    1(총 168) 화일명: EXAMPLE2.GEN  모드: [ 테이블 ]  준영역: 4(총 295K)
[ 겹쳐쓰기 ] [보통문자][ Ragged Align ]
<......+.......+.......+.......+.......+.......+.......+.......+.......+.......
........1.........2.........3.........4.........5.........6.........7.........
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
.save a
제 1 장  서  론

제 1 절 연구의 목적

.save b
I
.return b
    llegitimacy 란 말은 혼외자
    를 뜻하는 말이다.
혼외자의 지위는 나라와  시대에
따라 다르나 서양에서는  일부일
처제의 혼인을 정당시하고  정당
한 혼인에 의하지 않는 성결합을
을 죄악으로 보고   죄있는 결합
의 죄없는 과실  인 혼외자를 자
별하기 시작하였다
  Humanism

                  사상은 혼외자
의 법적 지위향상에 관심을 기울

 제 2 절  연구의 범위와 방법

.save c
혼
.return c
    외자의 현황을  발생 원인과
    그 실태면에서 외국과  우리
나라의 경우로 나누어  고찰하고
외국의 혼외자  입법을 고찰함에
있어서  주제를 중심
어 보기로 한다.

             에 대해 혼외자
            부양, 상속 등으로
나누어 고찰하고, 혼외자에 관한
특수문제로 인공수정자와 혼외자
의 관계 등을 살펴보기로 한다.
.return a
.indent 33
제 2 장   혼외자의 발생 원인과
리 등 카톨릭 국가에서 이혼법이
성립되기 전에 찾아볼 수 있었다.

2. 우리나라에 있어 혼외자의
   발생 원인

1 )  법률상 원인

.save d
법
.return d
    률이 명문으로 금지하고  있
    는 조항을 위반함으로서  혼
외자가 발생되는  경우는 1 약혼
중에 출생한 자,2 사실혼 부부간
에 발생한 자,3 첩 등 법률상 처
가 아닌 여자가 출생한 자, 4 혼
인에 의해 출생한 자 중
의해 그의 친생 확
   5 중혼에서
```

〔보기 2〕 세이브와 리턴명령을 이용한 예

제 1 장 서 론

제 1 절 연구의 목적

Illegitimacy 란 말은 혼외자를 뜻하는 말이다.
혼외자의 지위는 나라와 시대에 따라 다르나 서양에서는 일부일처제의 혼인을 정당시하고 정당한 혼인에 의하지 않는 성결합을 을 죄악으로 보고 죄있는 결합의 죄없는 과실인 혼외자를 차별하기 시작하였다. 그러나 근대이후 Humanism 에 입각한 인간존중과 만민평등의 사상은 혼외자의 법적 지위향상에 관심을 기울이게 되었다. 20세기에 들어와서 세계제국은 혼외자의 실질적 평등을 보장하기 위해 노력해왔는데 모든 제국이 이법으로 혼외자의 법적 지위 보장을 요구하고 있다. 오늘날 전 세계를 통하여 해마다 격증하고 있는 미혼모와 혼외자 문제는 각국마다 심각한

문제를 안겨주고 있는데, 각국은 혼외자법에 중요한 관심을 가지고 혼외자법의 중심과제를 [혼외자의 사회적, 법적 평등의 보장]에 두는 것 같다. 그러한바 현실의 혼외자 실태를 살펴보고, 외국의 혼외자입법을 비교 고찰함으로서 혼외자의 법적 지위를 연구하는데 그 목적을 두고자 한다.

제 2 절 연구의 범위와 방법

혼외자의 현황을 발생 원인과 그 실태면에서 외국과 우리나라의 경우로 나누어 고찰하고 외국의 혼외자 입법을 고찰함에 있어서 주제를 중심으로 비교하여 보기로 한다. 그리고 이어 우리 혼외자 입법제에 대해 혼외자와 부권관계, 부양, 상속 등으로 나누어 고찰하고, 혼외자에 관한 특수문제로 인공수정자와 혼외자의 관계 등을 살펴보기로 한다.

제 2 장 혼외자의 발생 원인과 실태

제 1 절 혼외자의 발생 원인

1. 일반적 원인

i) 사회과학 사전에서는 미국에서의 혼외자의 발생 원인을 1920 년대에는 비도덕성(immorality), 심신박약병(mental deficiency)을 들고 있고, 1930 년대에는 가정 파탄(broken home),빈곤 (poverty), 와해된 이웃관계 (disorganized neighborhood)를 혼외자의 발생 원인으로 보고 있다.
ii) Krause는 미국에 4인의 흑인 아기중 1 인은 혼외자라 하여 혼외자를 가난의 문화(culture of poverty) 또는 니그로의 문화 (culture of negro)의 산물로 보고 있다.
iii) 또한 사회과학 사전에서는 1940년대와 1950년대 초반의 혼

외자의 발생 원인으로 감정 불안 (emotional disturbance)을 강조하고 있고, 1960 년대에 와서는 여러 나라에서 증대되고 있는 혼외자의 통계를 보고 이른바 "cross-culture analyses" 를 하기 시작하였다는 것이며 이러한 연구 태도는 혼외자에 대한 사회 문화적 이론에 이끌려졌다는 것이다. 이밖에 경제 융성기에도 혼외자의 발생률이 많으며 전쟁 직후에도 이러한 현상이 일어난다고 한다.

iv) 교육정도가 낮을 수록 미혼모가 많다는 것으로 보아 교육의 중요성이 강조된다. 또한 미국에서는 상당수에 해당하는 미혼모가 모에 의해 지배되는 가정에서 나온다 한다.
v) Goode는 혼외자의 발생 원인으로 1 합의에 의한 결합, 2 제도 화된 축첩 (중국과 일본), 3 하층 계급의 비적출, 4 약혼중의 출산 등 14 가지로 분류하고 있다.

혼외자의 발생 원인은 법적 측면인 혼인법과 이혼법에도 깊은 연관이 있는데, 법률혼만 인정하고 있는 국가에서 사실혼이 성행

된다면 그 부모로부터 출생한 자녀는 혼외자라는 운명에 처하게 될 것이고 이혼이 금지되어 있는 국가에서는 별거 상태에서 사실혼에 들어가는 남녀가 많아질 것이므로 여기서 출생된 자녀는 혼외자가 될 것이다.

이런 측면에서 법률혼주의는 혼외자 발생의 큰 원인이 된다고 말할 수 있다. 이러한 예는 이태리 등 카톨릭 국가에서 이혼법이 성립되기 전에 찾아볼 수 있었다.

2. 우리나라에 있어 혼외자의 발생 원인

1) 법률상 원인

법률이 명문으로 금지하고 있는 조항을 위반함으로서 혼외자가 발생되는 경우는 1 약혼중에 출생한 자,2 사실혼 부부간에 발생한 자,3 첩 등 법률상 처가 아닌 여자가 출생한 자, 4 혼인에 의해 출생한 자 중 판별에 의해 그의 친생 확정이 부인되는 자, 5 중혼에서 출생한 자, 6 근친간에서 출생한 자, 7 사통관계에 의해 출생한 자, 8 무혼의 당사자가 출생한 자 등이 있다. 우리 민법은 사실혼주의에서 법률혼주의로 전환하였기 때문에 사

실혼자인 혼외자가 많이 있고, 또 동성동본의 금혼 규정에 따라 금혼 범위가 세계에서 유례가 없을 정도로 광범하기 때문에 발생되는 혼외자가 상당수에 이르고 있다.

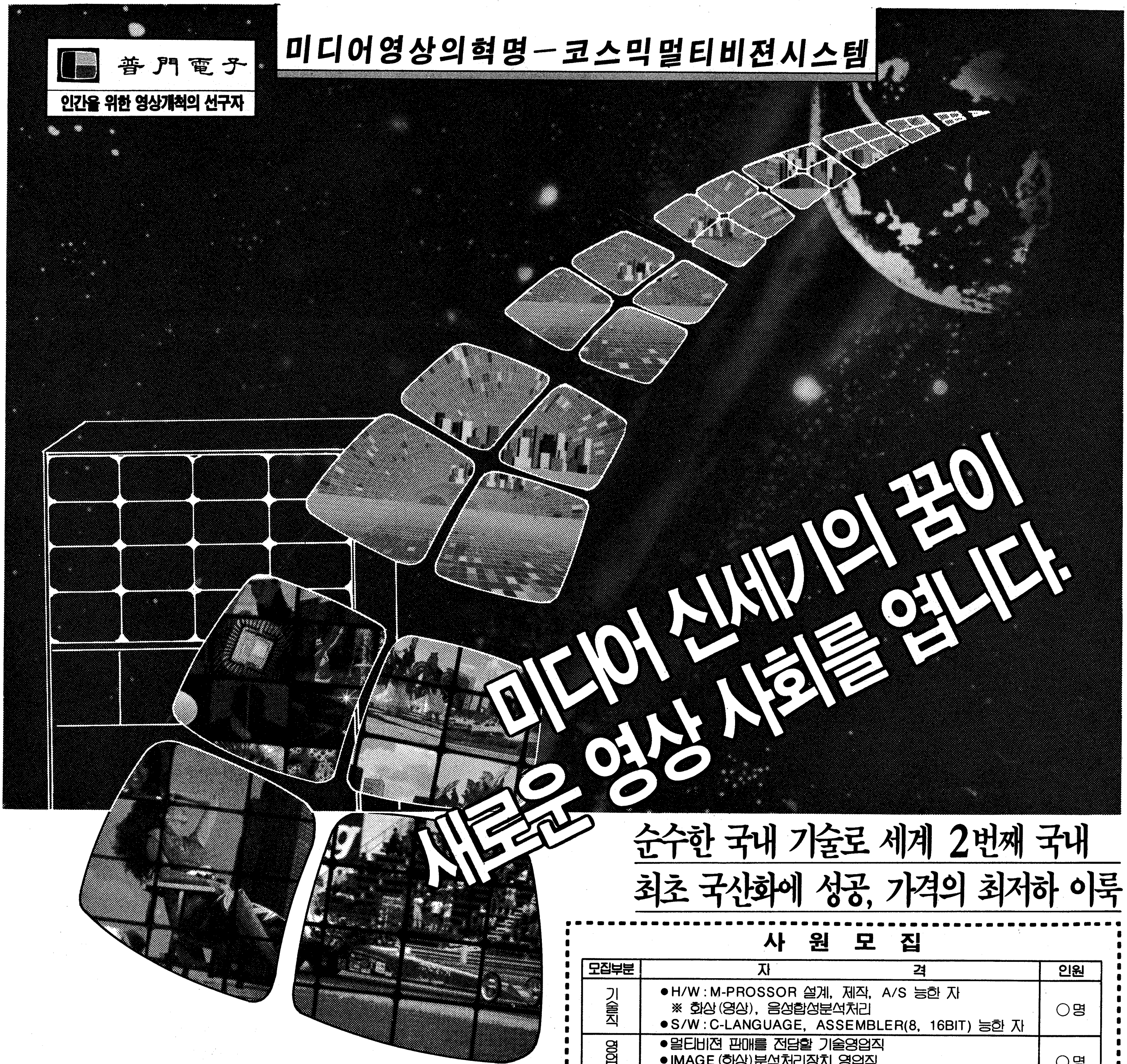
2) 법률 이외의 원인

사회복지회인 홀트아동복지회는 혼외자의 발생 원인을 다음과 같이 분류하고 있다.

이제까지 복수컬럼 인쇄방식에 대해 간단히 설명했다. 초보자들을 위해서 아주 간단한 예제를 들어 설명했지만 사용자들이 직접 연구해 본다면 더 복잡한 모양도 간단히 할 수 있다.

사실상 신문이나 잡지, 회지 등을 만들기 위해서는 기본적으로 제공되는 문자로는 표현가능한 정도가 한정되어 있다는 것은 당연한 사실이다. 그러므로 그래픽을 이용한 그림의 사용, 문자의 도안, 적당한 모양의 그래픽 문자 등은 절대적으로 필요하다.

마지막으로 다음번에는 보석글에서 위에서 말한 것들을 해결할 수 있는 방법을 소개할 것을 약속 드리면서 필자의 글을 마치기로 한다.



순수한 국내 기술로 세계 2번째 국내
최초 국산화에 성공, 가격의 최저하 이룩

사 원 모 집

모집부분	자 격	인원
기술직	●H/W : M-PROSSOR 설계, 제작, A/S 능한 자 ※ 화상(영상), 음성합성분석처리 ●S/W : C-LANGUAGE, ASSEMBLER(8, 16BIT) 능한 자	○명
영업직	●멀티비젼 판매를 전담할 기술영업직 ●IMAGE (화상) 분석처리장치 영업직 ※ F/A, M/A 고가장비임	○명

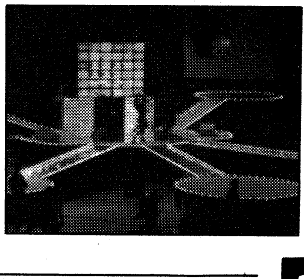
※ 각 부문 공히 경력사원 우대, 책임 운영할 경영자·중견간부 환영
※ 공히 군필자

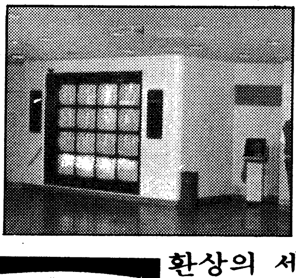
- 제출서류 : 이력서, 자기소개서
- 제출기간 : 공고일부터 (선착순)
- 전형방법 : 서류전형 합격에 의하여 개별면접 및 실시일정 통보
- 제출된 서류는 일체 반환치 않음 (서류는 우편으로 우송 및 연락처 명시)

NEW

멀티 비젼









■ 코스믹 아트비젼의 활용

- 기업홍보 ● 전시장 디스플레이 ● 방송국콘서트홀 무대장치
- 대중집회 행사 ● 호텔쇼핑센터, 백화점 ● 공공시설 ● 각종정보
- 고 부가가치 수출품

새로운 뉴모드 보급형 개발완료 납품준비 임대성업중

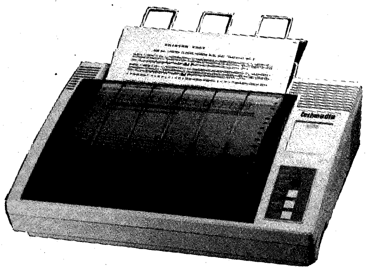
환상의 세계로 !

COSMIC COLOR
BOMUN ELECTRONICS CO.,

576-21, Eungam-Dong, Eunpung-Gu, Seoul, Korea
TEL : 302-3116, 304-5780

신제품

저가형 9핀 프린터
techmedia-1509



본 제품은 늘어나는 PC 사용자와 정보이용자가 고가의 가격때문에 프린터를 구입하지 못하고 여러가지 정보를 단지 화면상으로만 보아야만 했던 문제를 해결한 9핀 도트매트릭스형 프린터이다.

명조체와 고딕체문자는 물론 영문과 한글까지 지원하고 있는 10만원대의 저가형으로 일반 PC사용자에게 적합하다.

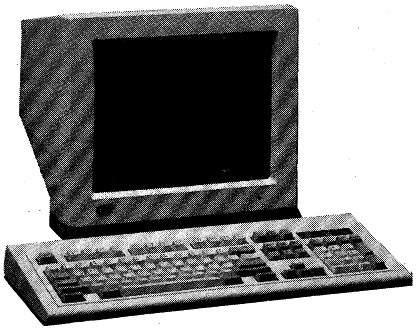
인자속도는 영문 135CPS, 한글 50CPS의 속도로 다소 늦은감이 있지만 각종 한글코드 선택이 가능해 한글 인쇄에 어려움이 없으며 다양한 인자방식을 갖추고 있다. 일반 용지도 사용할 수 있는 이 프린터에 시리얼포트(RS-232C)를 부착할 경우 데이터전송을 75~9600 보오까지 할 수 있다. 또 이 프린터에는 8KB의 버퍼가 내장되어 있어 시스템을 효율적으로 사용할 수 있고, 원본외로 2장을 더 프린트할 수 있는 카피기능이 추가되어 있다.

주요 규격은 다음과 같다.

- 인자방식 : 도트 매트릭스방식
- 인자헤드 : 9핀
- 인자속도 : 영문-135 CPS
한글-50 CPS
- 해당인자수 : 영문-80자(축소 132자)
한글-40자(축소 80자)
- 인터페이스 : 8비트 센트로닉스
(시리얼은 선택사양)
- 글자구성 : 영문- 9×9
한글-13×15
- 버퍼 : 8KB
- 용지폭 : 10인치
- 카피 : 원본외 2장

문의처 : 중앙시스템(주)
전 화 : (02)586-0141

한자/한글 터미널
HQ 3578



IBM 3270 모드의 단말기이며 IBM 5578 터미널과 호환인 한글/한자 터미널 HQ 3578 DFT(Distributed Function Terminal) 방식을 채택하였으며 IBM 3274/3174 제어장치와 큐닉스 한글/한자 프린터인 QP 3587을 연결하여 사용할 수 있다.

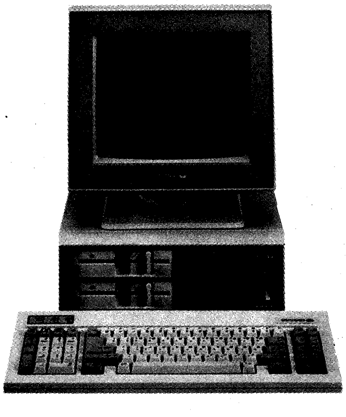
이 제품의 특징은 Multi-session 기능을 채택하여 한 단말기에서 5개까지의 세션을 키보드조작으로 바꿀 수 있다는 것이다.

주요기능은 다음과 같다.

- 3270 Data Stream 지원
- Extended Data Stream 지원
- 폐선(Grid Line), 혼용구간(Mixed Field) 지원
- 101키 키보드 사용
- 커서 선택 사용
- 14인치 평면모니터 사용
- IBM 코드와 KS 2바이트 완성형 코드 사용

문의처 : (주)큐닉스
전 화 : (02)783-4301

16비트 교육용컴퓨터
CPC-4000LC



교육용컴퓨터로 16비트가 선정되면서 다기능에 저가격대 컴퓨터의 보급이 확산되고 있다. CPC-4000LC는 기존보급형 PC에서는 플로피디스크가 한대뿐이었던 것을 두대로, 메인메모

리 256KB를 512KB로 확장하여 응용 소프트웨어의 활용도를 높여주고 있다.

CPU로는 10MHz의 i8088-1을 사용, 처리속도를 높여줌으로써 업무수행도 가능하며, 컬러모드가 내장되어 있어 별도의 컬러보드없이 컬러모니터를 사용할 수 있도록 하였고, 영문모드에서는 16색 지원도 가능하다. 시스템의 속도를 표시하는 LED가 전면에 부착되어 있어 사용자의 필요성에 따라 선택 사용할 수도 있다.

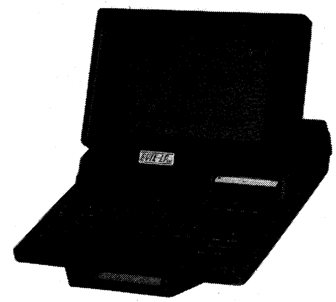
한글방식의 지원으로는 DH한글, 한글MS-DOS, KS 표준한글 등 세가지중 선택할 수 있도록 하여 다양한 프로그램을 사용할 수 있다.

- CPU : i8088-1 또는 EQUIV
- 클럭속도 : 4.77 / 8 / 10MHz
- RAM : 512KB(640KB 확장가능)
- 보조기억장치 : FDD-0.5MB×2(5.25인치)
  HDD-20MB (옵션)
- 포트 : 패러럴-1개
  시리얼-선택 사양
- 확장슬롯 : 4개
- 비디오모드 : MDA, HGC, CGA, PLANTRONICS
- 내장S / W : GWBASIC, 영문 MS-DOS, PC-TUTOR, 천리안II 에뮬레이터, 워드프로세서, MUG(MENU GENERATOR)

문의처 : 대우전자
전 화 : (02)751-1537

랩탑형 PC/AT
BYTE-LP 286

국내에서도 활성화되고 있는 랩탑 컴퓨터 시장에 활기를 불어 넣어준 이 기종은 CPU로 12MHz의 80286을 사용한 휴대용 컴퓨터로서, 메인메모리를 640KB에서 4MB까지 확장할 수 있고, 주변기억장치로 3.5인치 플로피디스크 1개와 하드디스크 1개를 탑재하고 있어 한글, 한자 등과 다양한 소프트웨어를 지원할 수 있다.



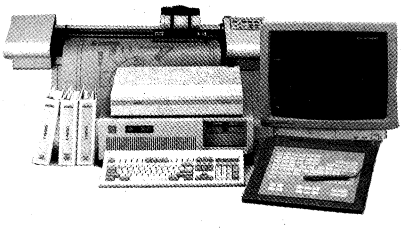
또한 가볍고 작아서 사용하기 편리하며 콤팩트하게 디자인되어 있는 이 기종은 특히 정전시 배터리 백업이 자동으로 되어 있어 1시간 이상 사용이 가능하다.

모니터는 LCD Backlight Display로서 밝은 곳이나 어두운 곳에서도 선명하게 볼 수 있으며 CGA와 HGC를 지원한다. 주요 규격은 다음과 같다.

- CPU : i80286-10(12MHz)
- 메인메모리 : 1MB
- FDD : 1.44MB(3.5인치)
- HDD : 20 또는 40MB
- 패러럴포트 : 1개
- 시리얼포트 : 2개
- 외부연결 : 16비트 버스 1개
- 비디오모드 : HGC(720×350), CGA (640×400)
- 리얼타임클럭 내장

문의처 : 바이트슈퍼
전 화 : (02)272-1144

고성능의 워크스테이션
CADRA-Ⅱ 시스템



워크스테이션의 빠른 처리속도와 PC의 저렴한 가격을 함께 모아놓은 CADRA-Ⅱ 시스템은 기존 PC로 CAD를 사용할 때 문제가 되었던 속도와 메모리문제를 해결한 시스템이다.

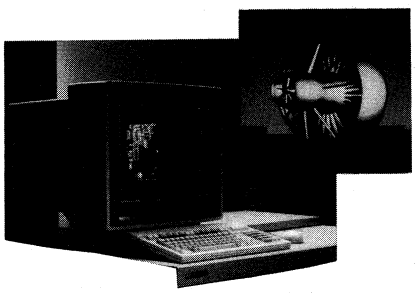
이 제품의 구성은 PC와 Acclaim Unit로 되어 있고, 서로간에는 패러럴로 인터페이스되어 CAD 운용시에는 80286 / 80386 CPU가 입출력에 활용되므로 3개의 CPU가 동시에 사용된다. 따라서 10MIPS라는 속도를 낼 수 있다.

주요기능으로 64개의 좌표축설정과 부분적인 드로잉, 멀티윈도우, 멀티스테이션 구성을 위한 네트워크, CADM 시스템과 인터페이스기능, 그리고 드로잉과 동시에 플로팅을 하는 등 PC 시스템에서 볼 수 없었던 다양한 기능을 갖고 있다.

- CPU : 모토롤러 68020
  –속 도 : 4MIPS
  –메모리 : 4MB RAM
- 그래픽 CPU : AMD 29116(8MIPS)
- 그래픽 가속장치 : Adra 커스텀게이트 어레이
- 모니터인터페이스 : EGA, VGA, IBM 8514 호환코넥터

문의처 : (주)어플라이드엔지니어링
전 화 : (02)679-7691

MC68000 채택한 워크스테이션
시리즈 300



HP 9000 시리즈 300 워크스테이션은 현재 필요로하는 성능뿐만 아니라, 안정된 구조, 구성 가능성, 확장 선택기능 및 호환성을 통하여 향후 확장에 대비한 제품으로 MC68000 구조를 채택하고 있다. OS로는 산업표준 UNIX OS에 버클리와 HP 자체가 개선한 기능들을 첨가하여 만든 HP-UX를 사

용하고 있으며, 명령어, 유틸리티, 컴파일러, 네트워킹, 그래픽라이브러리, 데이터베이스, X 윈도우 등 모든 분야에 걸쳐 한국표준코드를 지원하고 있다.

또한 계측기제어 및 일반과학언어로서 HP BASIC도 사용 가능하며, RS-232C, HP-IB(IEEE488), LAN 인터페이스 및 DMA가 내장되어 있다.

특히 지원시스템으로서 다음의 특성을 포함하고 있다.

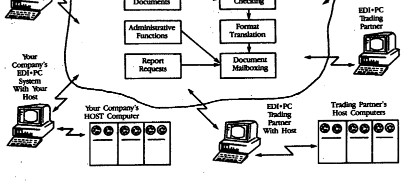
- 2~8 MIPS 처리능력으로 인한 계산 집약적 업무의 부담 경감
- I/O 확장성을 통한 다중업체 환경 연결용 통신 게이트웨이 역할
- 디스크리스 노드 및 주변장치 공유를 통한 각 작업 클러스터의 경비절감
- 화일서버기능을 통한 프로그램과 데이터 개정제어 및 중앙 액세스 허용

다음은 모델별 규격을 표시한 것이다.

모 델 명	330	360	370
CPU	MC68020	MC68030	MC68030
클럭속도	16.7MHz	25MHz	33MHz
co-CPU		MC6882	MC6882
주 기 억 용 량	4MB (Max 8MB)	4MB (Max 12MB)	
연산속도	2MIPS (97k flpos)	5MIPS (400k flpos)	8MIPS (730k flops)

문의처 : 삼성 휴렛팩커드
전 화 : (02)784-2666

전자식 서류교환 시스템 EDI*PC



GEIS(General Electric Information Services)에서 개발한 퍼스널컴퓨터용 EDI(전자식 서류교환 : Electronic Data Interchange) 변환 소프트웨어로 GEIS의 EDI Express 중앙시스템과 직접 연결하여 EDI 서비스를 이용할 수 있다.

상업문서들인 제품명세서, 주문서, 선하증권, 신용장 등이 EDI 표준양식으로 만들어져 있어 EDI Express 시스템을 통하면 어느 나라의 거래선들과도 손쉽게 데이터를 교환할 수 있다.

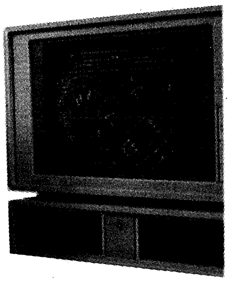
EDI*PC의 주요기능으로는 문서의 화면입력, EDI 표준화일로 자동변환, 대형 컴퓨터와의 접속, 문서의 프린트와 송수신, 보고서 작성과 문서관리, 송수신 시간 지정 및 다른 프로그램과의 접속기능 등이 있다.

EDI*PC 이용시 필요한 하드웨어의 구성은 다음과 같다.

- IBM PC / XT / AT 또는 호환기종
- 주기억장치 : 512K 이상
- 보조기억장치 : 10M 이상
- Hayes Smart Modem 또는 호환모뎀
- 132컬럼 프린터
- PC DOS V2.1 이상

문의처 : 데이타 매니지멘트(주)
전 화 : (02)782-2229

3차원의 복잡한 금형가공용 S/W EUKLID



금형가공에 있어 복잡한 부분까지 소프트웨어로 처리할 수 있는 EUKLID는 다양한 표면모듈의 제공과 표면간섭에 의한 복잡한 혼합을 처리할 수 있다.

또한 작업처리시간의 자동계산과 다양한 포스트프로세서의 제공으로 2½축에서 5축까지 동시 제어가 가능하고 체적과 모멘트계산기능도 갖추고 있다.

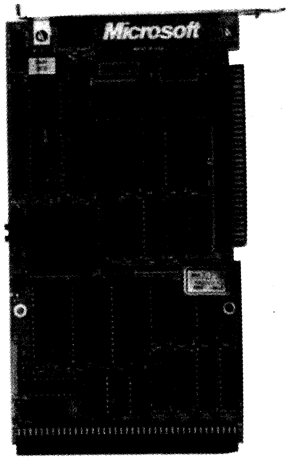
이 소프트웨어의 사용은 단조나 프레스, 사출금형가공 등 금형제작에 관련된 모든 업무에 적용시킬 수 있다.

EUKLID 운영시 필요한 하드웨어로는 다음과 같다.

- 시스템 : APOLO DOMAIN, VAX
- 플로터 : CALCOMP, HP
- 프린터 : 레이저프린터
- 디지타이저 : SUMMAGRAPHICS

문의처 : (주)다산시스템
전 화 : (02)757-7877

XT를 AT로 향상시키는 MACH 20



마이크로소프트 MACH 20은 성능향상 어댑터로서 IBM PC / XT 및 호환기종 사용자에게 응용소프트웨어 사용시 수행속도의 향상과 MACH 20 Memory Plus의 추가사용으로 3.5MB의 중첩확장메모리를 사용할 수 있는 제품이다.

컴퓨터시스템보드상의 8088 프로세서를 MACH 20에 포함되어 있는 80286 프로세서로 대체함으로써 8MHz의 속도를 얻을 수 있고, 성능향상 중첩확장메모리(EEMS)를 사용하는 응용프로그램은 8MHz의 PC에서 작동하는 것

만큼 빨라 더 많은 양의 정보를 신속히 처리할 수 있다.

MACH 20 보드의 연결단자는 마우스 및 그래픽 입력장치와 호환성을 유지하고 마이크로소프트 윈도우 및 OS/2 응용 소프트웨어에서 제공되는 기능 전체를 이용할 수 있는 처리속도를 제공해 준다.

또한 MACH 20에는 MACH 20 Disk Plus 다중매개 디스크제어를 연결할 수 있는 단자가 있어 1.2MB 5.25인치, 720KB 또는 1.44MB 3.5인치 디스크 드라이브를 사용할 수 있도록 지원하고 있다.

- CPU : 80286
- 클럭속도 : 8MHz
- 포트 : 마우스용
- MACH 20 Memory Plus(옵션)
- 다중매체디스크제어기 : MACH 20 Disk Plus(옵션)

문의처 : (주)마이크로소프트
전 화 : (02)782-3011

이미지프로세싱이 가능한 RISC 시스템 SYS 8864



모토롤러의 32비트 RISC CPU MC88100 을 사용한 CPU보드로서 MVME188을 탑재한 이 제품은 CPU를 1개에서 4개까지 확장하여 최고 60+ MIPS의 속도를 낼 수 있으며, 최대 250개 비동기 시리얼포트를 구성할 수 있다. 기존의 VME시스템에서 CPU보드를 교환함으로써 경제적인 RISC 시스템을 구축할 수 있는 이 시스템은 최대 2.4G바이트의 디스크용량을 구성할 수 있어 시뮬레이션, 이미지프로세싱, AI머신 및 고속멀티유저용 등에 사용할 수 있다.

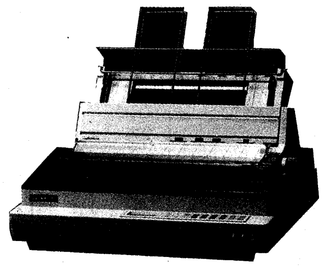
OS로는 UNIX SYSTEM V R3.2를 사용하며 RFS, X-window, C, FORTRAN, PASCAL 등과 Informix, ORACLE, Unify 등의 소프트웨어 운영이 가능하다. 또한 개발용 패키지로 VMEexec 등이 지원되고 있다.

이 시스템의 주요 구성은 다음과 같다.

- CPU : 모토롤러 MC88100
- 속도 : 17MIPS(최고 60+MIPS)
- 캐시메모리 : 128KB
- 메인메모리 : 16MB(최대 64KB)
- 시리얼포트 : 2개(최대 RS-232C 250개 이상)
- 디스크드라이브 : 1대(최대 SCSI 윈체스터 4대)
- 테이프드라이브 : 150MB
- 슬롯 : 20개

문의처 : M-시스템
전 화 : (02)833-5548

저소음 고속프린터 PHOENIX-550



본 제품은 55dB이하로 소음을 감소시키고 550 CPS의 속도로 업무의 처리속도를 향상시킨 프린터이다.

이 프린터는 전면에 프론트 스위치 패널(Front Switch Pannel)을 부착하여 유저들이 사용을 용이하게 할 수 있도록 되어 있다. 또 IBM PS/55, IBM 5540, KS 표준완성형, IBM 2바이트, KS 2바이트, N바이트 등의 다양한 한글코드를 처리할 수 있어 수퍼미니, PC 등 기종에 관계없이 접속이 가능하다.

인자헤드로는 24핀 도트 매트릭스 임팩트방식을 채용하였고, 낱장 연속장치를 소프트웨어 전환으로 사용할 수 있다.

또한 로직보드는 PAL 및 LSI 칩을 사용하여 메모리 및 스피드를 대폭 확장하였으며, 여러가지 해당하는 메인 프로그램을 수정하지 않고 소프트웨어를 조작하므로써 프린팅이 가능하도록 되어 있다.

기타 규격은 다음과 같다.

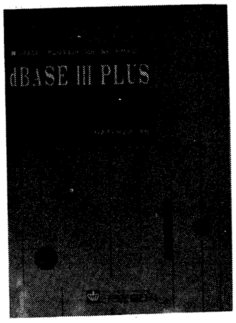
- 문자종류 : 한글(조합형(Full), 2350자(완성형)), 한자(4888자 완성형), 영문(Full ASCII)
- 인터페이스 : 센트로닉스 페러럴, RS-232C(옵션)
- 헤드수명 : 3억 DOT/PIN 이상
- 개행피치 및 개행속도 : 1/6", 1/120×n"(n≦255), 51PS~91PS

문의처 : 동양시스템산업(주)
전 화 : (02)587-2541~8

신간안내

dBASEⅢPLUS 핸드북

이준구 · 이승구 편역 / 크라운출판사 발행 / 정가 9,000원 / 608면 / 4×6배판



이 책은 dBASE시리즈 개발 이래 가장 많이 사용되고 있는 dBASEⅢPLUS를 새로이 배우려는 사람들은 물론, 좀 더 고급 방식으로 사용하고자 하는 사람들을 위해 매우 요긴한 내용들을 수록하고 있다.

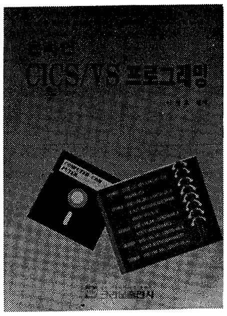
책의 특징을 보면 모든 명령어가 예제와 함께 수록되어 있고, 별도의 인덱스 없이도 쉽게 찾아볼 수 있도록 명령어가 기능별로 분류되어 수록되어 있으며 또한 dBASEⅢPLUS사용의 극치라 할 수 있는 네트워크 상에서 dBASEⅢPLUS와 LAN을 이용하여 멀티유저를 구현하는 방법도 설명되고 있다.

책 구성은 전체가 24장으로 되어 있으며 제1장은 dBASEⅢ-PLUS 의 사용환경과 설치시 발생되는 오류해결법, 제 2장은 화일명과 화일 구분법, 3, 4, 5장은 화일작성과 변경 및 처리에 관한 명령어, 6, 7, 8, 9장은 데이터의 추가, 편집, 표시, 위치 지정에 관한 명령어, 10장은 메모리 변수, 11장은 파라미터 제어, 12장은 프로그램 인터페이스, 13장은 대용처리, 14장은 프로그램 제어, 15장은 디버깅 명령어, 16장은 사용자 지원을 위해 새로 부가된 기능, 17장은 네트워킹, 18장은 각종 연산자, 19장에서 24장까지는 데이터베이스, 수치, 스트링, 날짜, 환경, 네트워크에 관련된 각종 함수에 대하여 설명되어 있다.

한편 부록에는 dBASEⅢPLUS로 작성된 프로그램 코드를 압축해 주는 기능을 가진 DBC와 dBASE링커인 DBL, 그리고 통신망에서의 dBASEⅢPLUS 사용법 등이 수록되어 있다.

온라인 CICS/VS 프로그래밍

이청호 편저 / 크라운출판사 발행 / 정가 5,500원 / 259면 / 4×6배판



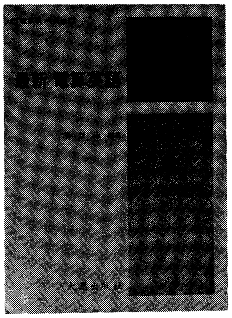
CICS / VS는 IBM사에서 개발한 온라인지원 S / W이다. 정보화사회라고 일컬어지는 현대는 대부분의 사회활동이 정보를 토대로 이루어지고 있으며, 컴퓨터 산업의 발전에 힘입어 대부분의 업무처리와 정보의 저장이 컴퓨터에 의해 이루어지고 있다. 따라서 다양하고 정확한 정보를 신속하게 획득하고 활용하는 것이 요구됨에 따라 CICS / VS가 개발되었다. CICS / VS는 처음에는 어셈블러와 함께 매크로 형태로 사용되었으나, 점차 일반 프로그래밍 언어와 함께 사용될 수 있는 코맨드레벨 인터페이스가 개발되어 프로그래머들은 CICS / VS에 대한 사용방법만 알면 온라인 프로그램을 작성할 수 있다.

이 책은 우선 COBOL에 대한 일반적 지식과 VSAM화일에 대한 기본적 지식을 가지고 있는 프로그래머들이 CICS / VS에 대한 지식을 쉽게 익힐 수 있도록 예제 중심으로 서술되어 있다. 이 책의 구성은 전체 11장과 부록으로 되어있는데 제 1장과 2장에서는 CICS / VS의 개요와 프로그래밍 개념에 대해서, 3장과 4장은 터미널과 프로그램 콘트롤에 대하여, 5, 6, 7장은 CICS / VS를 구성하고 있는 매니지먼트 모듈의 사용법에 관하여, 8, 9장은 BMS(Basic Mapping Support)의 기능과 사용법에 대하여 서술하였고, 10장은 프로그램의 디버깅과 테스트, 11장은 CICS에서 제공되는 시스템 태스크의 사용법에 대하여 설명되어 있다.

그리고 부록에서는 한글 HEXA 코드와 시스템 태스크의 종류와 용도, 명령문 요약표 등이 첨가되어 있다.

최신 전산영어

오세영 편저 / 대은출판사 발행 / 정가 6,000원 / 302면 / 4×6배판



정보처리 분야의 자료들은 대부분 영문으로 되어 있다. 더우기 정보처리 분야는 국제화되어 있고 영문자료를 접해야 할 기회가 늘어나고 있으며, 이런 상황하에서는 정보처리 분야의 영문 독해력이 절실히 요구되는 때라 하겠다.

이 책은 컴퓨터 초보자라도 컴퓨터 관련서적을 대할때 친숙해질 수 있도록 함과 동시에 정보처리기사, 기술사, 전산 전문직 취직시험 등을 대비하는 수험생중 정보처리 분야를 이해하려는 사람들에게 도움을 줄 수 있도록 되어 있다. 대학 · 전문대학 · 각종학교의 교과서 내지 참고서로도 사용될 수 있도록 엮어져 있으며, 최근에 등장하고 있는 새로운 용어와 분야까지

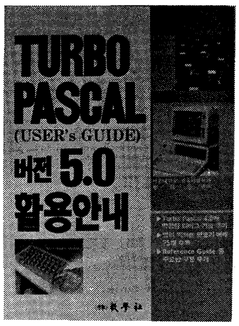
다루어져 있다.

이 책은 세부분으로 구성되어 있다. 제 1부는 기초편으로 전자계산학 전반에 걸친 영문을 10개 장에 걸쳐 주요항목별로 선정하고, 쉽게 익혀나갈 수 있도록 용어에 대한 자세한 해석을 곁들이고 있다. 제 2부는 응용편으로 다양하고 수준높은 문제를 주요항목별로 5개의 장에 집약, 수록되어 있다.

마지막으로 제 3부는 실전편인데 앞에서 배운 사항들을 복습하고 각종 시험에 대비하여 실전감각을 익히도록 하고, 과거 정보처리기사 시험에 출제되었던 문제들을 분석하여 전문해석, 빈칸채우기, 연결형, 선택형, 진위형, 정보처리 용어등을 유형별로 분류하여 수록하였다.

TURBO PASCAL (user's guide) 버전5.0 활용안내

황희융 편저 / 교학사 발행 /
정가 7,000원 / 570면 / 4×6배판



파스칼 언어는 1970년대초 스위스의 니클라우스빌쯔가 프로그래밍 교육을 목적으로 설계한 것인데, 이전의 프로그래밍 언어들을 바탕으로 만들어졌기 때문에 매우 체계적인 언어라고 볼 수 있다. 특히 구조화 프로그래밍(Structured Programming)을 효율적으로 구현할 수 있도록 되어 있으며, 엄격하면서도 매우 다양한 데이터형을 제공하고 있기 때문에 많은 사람들이 애용하고 있다.

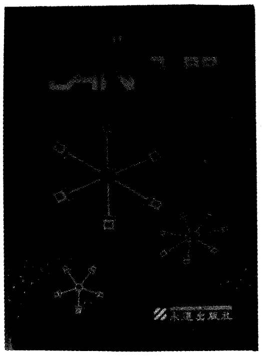
볼랜드사의 TURBO-PASCAL5. 0은 버전4.0에 비해 통합 디버깅 환경을 대폭 강화했다는 사실에서 많은 관심을 모으고 있다. 그중 통합디버거에는 프로그램의 논리적 오류를 쉽게 찾아내어 수정을 용이하게 할 수 있도록 되어있어 초보자들이 프로그램이나 알고리즘을 이해하는데 큰 도움을 줄 수 있다.

이 책의 제 2장 "터보파스칼 맛보기"와 2장의 부록편을 보면 통합 디버거의 사용법에 중점을 두어, 디버깅 방법이 단계적으로 설명되어 있어 쉽게 익힐 수 있을 것이다. 한편 이 책은 터보파스칼5.0의 User's Guide를 바탕으로 터보파스칼의 환경자체의 사용법에 특히 중점을 두어 번역되었으며 그외에도 사용자의 이해를 돕기 위하여 오버레이, 입출력, 프로시듀어와 함수목록, 코드표, 오류설명문 등은 Reference Guide에서도 뽑아서 함께 수록하고 있다.

이 책의 본문은 전체 11장과 부록으로 구성되어 있다.

PC통신시리즈 Ⅱ LAN 입문

묵현상외 1인 편저 / 영진출판사 발행 /
정가 7,500원 / 378면 / 4×6배판

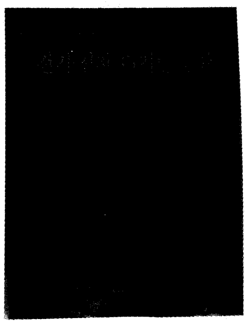


LAN입문은 PC데이터 통신의 기초와 각종 통신, 하드웨어 및 소프트웨어, 통신 프로토콜, 네트워크 아키텍쳐 등 PC통신 전반에 관한 사항과 광역 통신망(WAN)에 관하여 중점적으로 서술된 "해설 PC 통신 입문"에 이은 PC통신 시리즈의 후편이다. 이 책은 전편의 데이터통신 및 광역 통신망 이론에 기초하여 LAN(Local Area Network) 및 관계 소프트웨어, 인터네트워킹, 어플리케이션 등을 설명하고 있기 때문에 PC통신의 초보자들은 전·후편 모두를 읽어야 PC통신을 보다 체계적으로 이해할 수 있을 것이다.

이 책의 구성은 전체 6장과 부록으로 되어 있는데 제 1장과 2장에서는 LAN통신의 기초와 소프트웨어에 관하여, 제 3장과 4장에는 네트워크의 상호접속 및 구성에 관하여, 제 5장에는 통신 및 LAN의 응용에 관하여, 제 6장에는 PC 데이터통신 및 네트워크 운용에 있어서 발생되는 문제점들의 질문에 대한 응답의 형식으로 짜여져 있다. 부록에는 여러 예제프로그램을 통한 PC 통신의 프로그래밍 기법 해설과 국내 및 해외정보통신 서비스에 관한 내용이 수록되어 있다.

전기·전자·디지탈공학 =이론·실제=

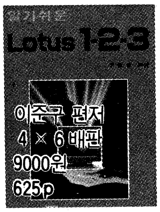
이영노외 1인 공저 /
기전연구사 발행 /
정가 8,000원 / 575면 / 4×6배판



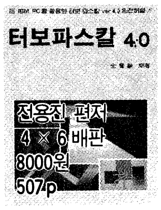
과학기술의 최첨단을 달리고 있는 정보산업육성은 산업사회가 대형화되고 정밀화됨에 따라서 무엇보다도 기술인력 확보를 필요로 하게 되었고, 이에 따라 기술교육을 위한 새로운 실험실습 교재가 필요하게 되었다.

이 책은 전기·전자공학의 대표적인 실험을 모아서 만든 기본적 교육을 위한 지도서로서 종합편으로 편찬된 것이다.

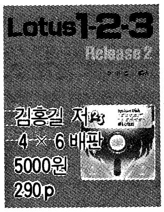
IBM PC 컴퓨터서적 베스트 셀러



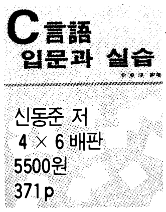
본 서적은 Lotus 123 Release 2의 최신 버젼으로 매뉴얼 방식을 탈피하여 초보자도 쉽게 이해할 수 있도록 Lotus 123의 제반기능을 알기 쉽게 설명하였으며 명령어도 예제와 함께 모든 버젼에 대한 해설과 높은 버젼으로 이행하는 법까지 기술하였다.



본서는 터보파스칼 4.0을 파스칼을 이용하는 독자들에게 쉽게 전달하고자 편역한 것으로 Turbo Pascal 4.0 사용방법 뿐만 아니라 Pascal로 프로그램을 개발하는데 필요한 내용도 함께다루고 있다. 또 버젼 3.0에서 4.0으로 수정하여 변환하는 방법까지 상세히 다루고 있다.



로터스 1-2-3의 기능에 자신도 모르게 감탄이 터져나온다. 그만큼 이 서적은 로터스 1-2-3을 알기 쉽게 설명하고 있다. 1-2-3의 막강한 힘과 기능을 이해하기 쉽게 설명하였다.



본서는 C언어를 초보자의 관점에 맞추어 입문에서 실습에 이르기까지의 제반 내용을 실제 프로그램을 예로하여 설명하고 있다. 상세한 설명이 C언어 입문과정 교재에 적합하다.



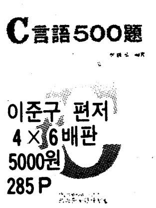
쿼트로는 로터스보다 탁월한 기능을 가진 환상적인 스프레드쉬트이다.

로터스 사용자는 물론 스프레드쉬트 프로그램을 처음 사용하는 초보자들도 쉽게 사용할 수 있도록 설명하였다. 입문·응용·실습까지!! (**신 간**)



IBM PC에서 가장 폭 넓게 사용되고 있는 패키지는 단연 1-2-3이다.

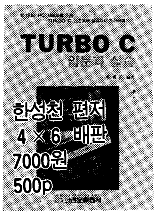
본서는 매뉴얼 형식을 탈피하여 초보자들도 쉽게 이해할 수 있도록 Lotus 1-2-3의 제반 기능을 실예와 기능위주로 설명하였다.



500개의 예제로 C 언어를 초보자의 사용자 관점에 맞추어 입문에서 실습에 이르기까지 C언어의 제반 내용을 실예로 들어 프로그램으로 설명하여 쉽게 이해할 수 있도록 하였다.



이책은 Prolog를 이용하여 프로그램을 작성할 때 응용프로그램을 효율적으로 개발할 수 있도록 80여개의 Tool 들과 40여개의 샘플 프로그램에서 Tool 사용방법을 예시하고 있다.



MS-DOS 환경에서 사용할 수 있는 컴파일러 중 Borland社에서 개발한 Turbo C 컴파일러를 사용해 쉽게 C 언어를 배울 수 있도록 편집하였으며 또한 C언어 전반과 Turbo C 모두를 배우는데 많은 도움을 줄것이다. Turbo C가 ANSI 표준을 따르고 있기 때문에 본서의 많은 개념들은 다른 컴파일러에도 그대로 적용될 수 있다. (**신 간**)



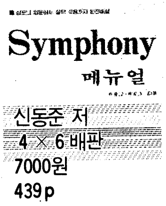
본서는 MS-DOS 컴파일러를 사용하는 프로그래머를 위한 참고서적이다.

내용은 컴파일러, 라이브러리 설치, 연산명령에 알기 쉽게 설명하고 있다. 터보 C와 MS-C와 유사 계열이다.

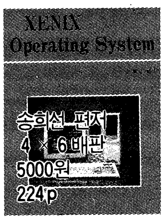


Turbo C 사용자 가이드로써 디스크의 내용과 하드디스크상의 설치법 등 개발환경 설정과 예제 프로그램의 컴파일과 실행 프로그램 작성방법 등을 소개하고 있다.

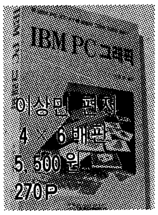
(**TURBO C Reference 5,000원**)



통합 소프트웨어로 수치계산에서부터 문서작성 데이터 관리 그래프 작성등 모든분야의 업무를 Symphony 하나만을 사용하여 처리할 수 있다. 최신 버젼 1.2 내용까지 수록하였다.



XENIX의 운영 체제(OS : operating system)를 배우도록 최신의 16/32비트의 XENIX 시스템의 OS에 대하여 설명하고 있는 기본 매뉴얼이다.



컴퓨터 언어 교육에 컴퓨터 그래픽스 활용이 날로 증가하는 추세이다.

본서는 컴퓨터 그래 픽스의 알고리즘을 담고 있다. 또 풍부한 예제 프로그램이 그래픽 유저에게 흥미를 줄 수 있도록 하였다. (**신 간**)



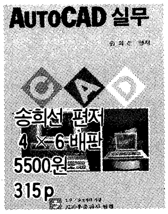
RM/COBOL은 Ryan-Mcfarland社에서 제작한 컴파일러를 사용하는 COBOL언어(표준 ANS COBOL 1974)로서 IBM-PC 호환기종에서 사용할 수 있으며 표준운영체제로 MS-DOS나 PC-DOS를 사용한다. (**신 간**)



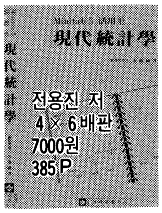
IBM PC/AT 퍼스널 컴퓨터와 DOS 3. XX Intel 80286 그리고 PC/AT BIOS에 대해서 상세히 다루었다. 주로 강조한 점은 DOS하에서 PC/AT를 효율적으로 사용할 수 있도록 하였다.



이책은 TorBo prolog 를 처음 대하는 사람도 쉽게 이해할 수 있도록 전문 용어를 풀어서 설명하고 예제에 대해서도 상세히 설명하였다. T-orBo prolog의 다양한 풍부한 예제는 실무에 많은 도움이 될 것이다.



본서는 Auto CAD에 관해 배우고자하는 사람을 위하여 알기 쉽게 Auto CAD를 다룰 수 있도록 설명한 마스터링 Auto CAD를 기본으로 기초에서 실예까지 함께 다루고 있다.



IBM PC을 활용하여 기초 통계학 과정에 있는 학생들을 위해 설계된 통계 프로그램 Minitab을 활용할 수 있도록 Minitab 사용방법을 설명하고 있다.



Micro soft사의 MS-DOS를 모든 버젼을 총망라하여 실제예와 사용상의 가이드를 싣고 있어 DOS명령어의 정확한 사용법과 확장된 사용법을 소개했다.

컴퓨터관련업체, 기업체 전산실 및 관공서, 컴퓨터 학원 납품 상담환영

전국 유명 서점 절찬판매중!!

첨단 기술서적 전문 출판!!

BOOK

크라운출판사

서울시 종로구 연건동 273-12
TEL : 745-0311, 0312, 0313
펙시밀리(FAX) 743-2930

●광장●

뭐든지 물어보세요

● 컴퓨터의 작동과정은 어떻게 이루어지는지…

컴퓨터에 전원을 넣으면 화면에 현재 메모리용량과 DOS의 버전등을 표시하고 드라이브를 가동시켜 프로그램을 입력할 수 있도록 프롬프트가 나오는데 이와같은 컴퓨터 작동 과정은 어떻게 이루어지는지요?

최준영 / 부산시 동래구 온천 2동 725-5 (19/6)

▲CPU와 RAM, ROM을 중심으로 작동

마이크로프로세서인 CPU와 전원이 들어오면서 컴퓨터가 작동할 수 있도록 제반사항을 관리하는 프로그램이 들어 있는 ROM과 사용자가 실행시키고자 하는 프로그램이나 데이터가 저장되는 RAM을 중심으로 작동이 시작됩니다. 물론 주변 칩들도 없어선 안되는 것들이지만 여기서는 이러한 것들을 중심으로 알아보도록 하겠습니다.

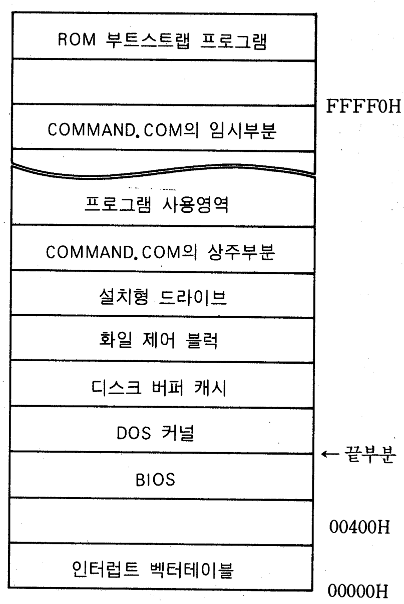
시스템에 전원을 넣거나 리세트 또는 Alt+Ctrl+Del 키에 의해 재시동하면 프로그램의 실행은 FFFF0H 번지에서 시작합니다. 이것은 MS-DOS 와 관계없는 8086계 마이크로프로세서의 특징으로 전체 어드레스 FFFFFH 중에서 ROM이 시작하는 번지를 뜻하는 것입니다. ROM에는 시스템이 동작할 수 있도록 동작의 제어나 상태 등을 표시하는 프로그램이 들어 있어 전원이 켜지면서 롬에 있는 프로그램이 실행됩니다. 이 프로그램에는 시스템 코드와 롬 부트스트랩 루틴으로 제어를 옮기는 점프명령이 포함되어 있고, 롬부트스트랩 프로그램은 메모리의 임의의 번지로 디스크의 첫번째 섹터(부트섹터: 디스크포맷에 관한 정보테이블)에 있는 디스크 부트스트랩 프로그램을 읽고 나서 제어권을 이 프로그램에 넘깁니다.

다시 이 디스크 부트스트랩 프로그램은 디스크에 있는 루트디렉토리의 첫번째 섹터에서 IO.SYS (본지 1, 6 월호 특집 참조)와 MSDOS.SYS (본지 3월호 특집 참조) 화일을 메모리내로 읽어들여 IO.SYS 화일로 제어를 넘기며, IO.SYS의 화일은 실제 두개의 모듈(BIOS와 SYSINIT)로 나뉘어져 메모리 재배치를 시작합니다. 즉, SYSINIT는 메모리의 크기를 결정하고 자신을 메모리의 최상위번지로 옮기고 MSDOS (이 화일을 DOS 커널이라고 함)와 BIOS를 재배치한 후 MS-DOS.SYS에 있는 초기화 코드를 호출함으로써 시스템에 관한 초기화작업을 시작합니다. 초기화작업이란 주변장치의 연결상태점검과 섹터크기 및 버퍼, 매개변수 블록 등을 만드는 것과 MS-DOS의 상표권 등을 표시하는 것을 말합니다.

다음 SYSINIT로 제어가 옮겨지고, CONFIG.SYS 화일 (MS-DOS 입문 참조)을 읽어들여 MS-DOS 환경을 사용자에게 맞추게 됩니다. 예를 들면 부가적인 하드웨어 디바이스 드라이버의 수, 디스크 버퍼의 수, 한번에 오픈할 수 있는 최대 화일의 수, 명령처리기의 화일명 등을 사용자가 지정할 수 있는 것입니다.

CONFIG.SYS 화일이 발견되면 이러한 내용이 메모리로 적재되어 모든 소문자는 대문자로, 명령의 수행은 한번에 한줄씩 해석되고, 메모리는 디스크버퍼 캐시와 내부 화일제어블록(FCB)을 위해 할당됩니다. 이 화일(CONFIG.SYS) 내에 기술된 임의의 디바이스드라이버는 메모리에서 INIT 모듈을 호출하여 초기화하고, 디바이스리스트내로 연결되면서 차례로 할당됩니다. 각 드라이버의 INIT 펑션은 SYSINIT에게 그 드라이버에서 사용될 메

〔그림 1〕 메모리 배치도



모리크기를 알려주게 됩니다. 이렇게 사용가능한 디바이스드라이버가 설치된 후 SYSINIT는 모든 화일핸들을 폐쇄하고 콘솔(CON), 프린터(PRN), 보조장치(AUX)등을 표준입출력, 표준에러, 표준리스트와 보조장치로서 사용할 수 있게 하는 것입니다.

마지막으로 SYSINIT는 명령해독기를 적재하는 MS-DOS EXEC 펑션이나 Shell(COMMAND.COM)을 호출하고, Shell이 읽혀지면 프롬프트가 화면에 출력되고 사용자가 명령어를 입력할 수 있도록 MS-DOS 작업준비가 완료되는 것이며 SYSINIT 모듈은 무시됩니다.

그림 1은 지금까지의 과정을 거쳐서 메모리배치가 완료된 상태를 나타낸 것입니다.

● DRHALO Ⅱ를 키보드만으로 작동할 수 있는지요?

DRHALO Ⅱ에서 마우스없이 키보드만으로 작동할 수 있는지요? 작동가능하다면 사용법을 가르쳐 주십시오.

정재홍/강원도 명주군 왕산면 왕산중학교

▲키보드뿐 아니라 포인팅 디바이스 등을 사용할 수 있어

DRHALO Ⅲ에서는 Genius마우스와 Summagraphics의 타블렛(tablet)만을 사용할 수 밖에 없지만 DRHALO Ⅱ에서는 마우스를 비롯하여 조이스틱, 디지타이저, 트랙볼, 키보드 등 모든 Pointing Device등을 사용할 수 있습니다.

DRHALO Ⅱ에서 입력장치로 키보드를 사용하려면 우선 키보드를 사용할 수 있는 사용환경을 만들어 주어야 합니다.

먼저 시스템에 SETUP.EXE화일이 있는 DRHALO 프로그램을 넣고 다음과 같이

```
SETUP ↵
```

하면 그래픽 카드를 선택하는 화면이 나옵니다. 여기서 독자가 소유하고 있는 그래픽카드와 일치하는 번호를 입력합니다. 그 다음 아래와 같이

```
Virtual page (Y or N)?
```

라는 메시지가 나오면 자신의 기종에 맞게 설정합니다.

그다음에 Pointing Device?라는 메시지가 나오면 키보드를 선택하고 선택번호를 입력합니다. 입력이 끝나면 Default Configuration?이라는 메시지가 나옵니다. 여기서는 y 혹은 n로 지정해 줍니다. 이것이 끝나면 화면이 DOS 상태로 복귀합니다.

여기서 다시 디렉토리를 살펴보면, INIT.BAT 화일과 DRHALO.CNF화일이 생겼을 것입니다. SETUP 작업이 끝나면 이 두 화일이 자동적으로 생성됩니다. 이로써 키보드를 사용할 수 있는 환경이 모두 조성된 것입니다.

그러면,

```
INIT ↵
```

을 입력한 다음 화면을 살펴보면 HALOKBDI라는 키보드 드라이버 화일에 대한 것이 나타날 것입니다. 그러면 다시 DRHALO를 입력합니다.

이제 화면을 살펴보면 그림을 그릴 수 있는 화면상태로 바뀌어져 있을 것입니다.

동작방법은 화면의 원하는 아이콘(Icon)으로 커서를 이동시켜 왼쪽 〈SHIFT〉키로 메뉴를 선택합니다. 그뒤 다시 화면으로 복귀하여 〈SHIFT〉 키와 화살표 키를 동시에 움직여 그림을 그려 나갑니다. 아이콘의 지우개와 같이 크기를 변환하여 사용할 때는 〈\〉키와 화살표 키를 동시에 이용하여 원하는 크기로 조정합니다. 이때 실행은 다시 〈SHIFT〉키와 화살표 키를 동시에 사용합니다.

여기서 〈SHIFT〉키는 마우스의 왼쪽 버튼, 〈\〉키는 오른쪽 버튼과 동일한 기능을 수행합니다.

● 한글코드 변환프로그램의 원리는

한글코드 변환프로그램은 어떠한 원리로 작성되는지요?

김용대/서울시 성북구 석관 1동 326-15 26/1

▲서로간에 해당되는 코드로 변환시켜 줍니다.

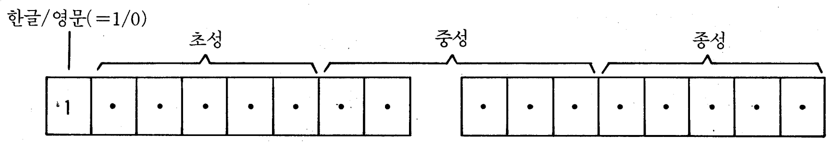
한글코드에는 N바이트, 3바이트, 2바이트 조합형, 2바이트 완성형, 7비트 2바이트 완성형(청계천 한글코드) 등이 있습니다. 이렇게 다양한 한글코드간의 상호교환을 가능하게 한 것이 한글코드변환프로그램입니다.

그런데 이 변환프로그램을 작성하려면 각 코드들의 특성에 대해 정확히 알고 있어야 하므로 실제로는 시중에 나와 있는 변환프로그램을 사용하는 것이 일반적입니다. 그러나 경우에 따라서는 프로그램속에 여러가지 한글을 수용할 수 있어야 하므로 그 원리를 살펴보는 것도 의미가 있겠습니다.

여기서는 현재 많이 사용하는 2바이트 조합형 한글코드의 변환방법에 대해서 설명하겠습니다. 2바이트 조합형 한글코드는 어떤 회사든 간에 **그림 2**와 같이 구성됩니다.

즉 초성, 중성, 종성을 각각 5비트로 묶어 나타내고 이들을 모아 하나의 코드를 형성합니다. 최상위 비트에는 1을 써넣어 한글임을 나타내는 표현구조를 갖고 있습니다. 이러한 표현방법은 각업체마다 동일하나 글자에 해당하는 코드에 있어서 서로 다르므로 그 코

〔그림 2〕 2바이트 조합형 한글코드 구조



〔그림 3〕 00100코드에 해당되는 각 회사별 인식표

	초성	중성	종성	초성	중성	종성	초성	중성	종성
00100 코드	ㄴ	ㅐ	ㄳ	ㄷ	ㅐ	ㄴ	ㄴ	ㅑ	ㄴ
	삼보, 현대, 대우, 쌍용, 유닉스			삼성			금성		

드를 파악하여 원하는 코드로 바꿔주어야 합니다.

예를 들어 5비트 00100 코드에 대한 인식표가 **그림 3**과 같이 각 회사마다 각기 다르므로 원하는 글자에 해당되는 코드를 찾아 이를 하나 하나 변환시켜 프로그램을 작성하면 이것이 곧 한글코드변환프로그램이 됩니다.

즉 만일 삼보한글코드를 삼성의 한글코드로 바꾸고자 한다면 먼저 한글코드 표시비트를 체크하고 그것이 1이면 그 다음 초성 5비트를 추출하여 코드표를 통해 대응하는 코드로 변환시켜 저장합니다. 마찬가지 방법으로 중성, 종성에 대해서도 적용하면 변환된 코드가 얻어지게 되는 것입니다.

조합형 한글을 2바이트 KS완성형 한글로 변환시키고자 할 경우에는 이와 달리 완성형 한글 테이블을 해당 조합형 한글의 인덱스(Index) 로 사용하여 변환하여야 하므로 조합형 간의 변환보다는 복잡합니다.

참고로 시중에 나와 있는 한글코드 변환프로그램으로는 청계천 등지에서 흔히 구할 수 있는 베이직 등으로 작성된 프로그램과 「마이크로소프트웨어」 지에서 게재한 여우꼬리나 카멜레온 등을 들 수 있습니다.

● 화일들을 알파벳 순으로 정리하려면

베이직 프로그래밍 작업을 하다보니 화일들이 많이 생기게 되었습니다. 현재 데이터 화일까지 합해서 300여개의 화일들이 무질서하게 기록되어 있어 특정 화일을 찾아보거나 프린팅하는데 큰 애로가 따르고 있습니다. 이 화일들을 영어의 알파벳 순으로 정리할 방법은 없는지요?

이종만 / 부산시 동래구 안락 1동 411-8

▲DOS유틸리티중SORT.EXE를 이용하면 쉽게 해결됩니다.

이런 종류의 문제는 DOS에 관한 문제입니다. 프로그램 화일이든 데이터 화일이든 디렉토리와 화일들은 DOS에 의해 공통적으로 관리되고 있기 때문입니다.

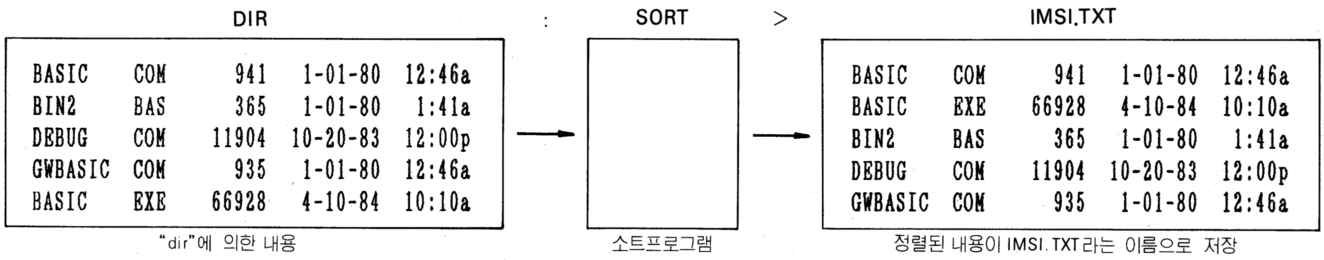
DOS유틸리티를 보면 SORT. EXE라는 화일을 볼 수 있습니다. 이 화일은 표준입력장치(주로 키보드)에서 데이터를 입력받아 데이터들을 오름차순, 혹은 내림차순으로 정렬시켜 주는 기능을 갖고 있습니다. 이렇게 정렬(SORT)된 데이터는 다시 표준출력장치(주로 화면)를 통해 출력되게 됩니다.

여기서 한가지 주의해야 할 점은 이 SORT명령은 DOS의 외부명령이므로 SORT.EXE란 화일이 꼭 있어야 한다는 점입니다.

```
DIR ¦ SORT > IMSI.TXT
```

위와 같이 실행하면 디렉토리의 화일들이 알파벳 순으로 정렬되어 IMSI.TXT라는 이름으로 저장됩니다. 여기서 '¦'기호는 파이프처리를 의미하는 것으로 이 기호를 사용하면 DOS에서 'dir' 명령에 의해 출력된 내용이 다시 SORT.EXE의 입력으로써 처리되어지는 것입니다. 그리고 정렬된 내용은 다시 '>' 기호(이 기호는 입출력 재지정, 즉 표준 입출력 디바이스들을사용자가 지정할 때 쓰는 기호입니다.)에 의해 IMSI.TXT란 이름으로 저장되는 원리입니다(**그림 4** 참조).

〔그림 4〕



경력 및 신입 사원모집

당사는 SNA 3270, BSC CARD, MUX 등의 통신용 CARD 제조 및 386을 비롯한 COMPUTER 제조, 판매회사로서, 금번 사세확장에 따라 최첨단기술에 사명감을 갖고 이끌어갈 유능하고 의욕적인 인재를 다음과 같이 모집합니다.

1. 모집부문 및 응시자격

모집부문		구분	인원	응 시 자 격
기 술 직		신입	00명	• PC에 대한 이해가 충분한 전문대학졸업 이상(예정)자
CE		신입	00명	• 전기, 전자, 기계과 출신 전문대학졸업(예정)자
연구개발		경력	0 명	• "C" 및 ASSEMBLER에 능통한자 • MICRO VAX 기종 유경험자
		신입	0 명	• 4년제 전자공학과 졸업(예정)자
관리직	총무	신입	0 명	• 관련학과 졸업(예정)자
	인사	신입	0 명	• 관련학과 졸업(예정)자
	자재	신입	0 명	• 관련학과 졸업(예정)자
	무역	신입	0 명	• 관련학과 졸업(예정)자

***공통사항**
- 군필 또는 면제자
- 해외여행에 결격사유가 없는자
- 해외 석사학위 소지자 우대
- 일어해독 및 해당분야 자격증 소지자 우대

2. 전형방법
- 1차 서류전형
- 2차 면접(개별통지)

3. 제출서류
- 자필이력서(사진부착, 연락처 명기)
- 자기소개서(취급기종 및 업무 중심으로 작성) : 경력사원에 한함
- 최종학교 전학년 성적증명서(신입사원에 한함)
- 최종학교 졸업증명서

4. 제 출 처
- 서울시 서초구 방배동 983—41호
  TEL: 581—1511 FAX: 588—2791

5. 제출기간
- 89년 8월 20일 도착날에 한함

6. 기 타 : 제출된 서류는 일체 반환치 않음

DIGITECH
DIGITAL TECHNOLOGY

대구 알파컴퓨터 시스템 개업

32BIT 80386 PC, 3270 한글통신 EMULATOR, ON LINE 통신용 BSC CARD, PC다중 통신용 MUX, 병렬처리 컴퓨터 ROM DISK DRIVE 등을 제조, 판매해 왔던 폐사는 알파컴퓨터 시스템이라는 이름으로 경북지역 총판대리점을, 5월 1일자로 개업하게 되었습니다. 우리는 이 개업을, 21C정보산업을 선도해 나가기 위한 하나의 자그마한 과정이라 여기고, 철저한 대 고객서비스 제일주의의 기치아래, 더욱 열심히 정진할 것입니다.

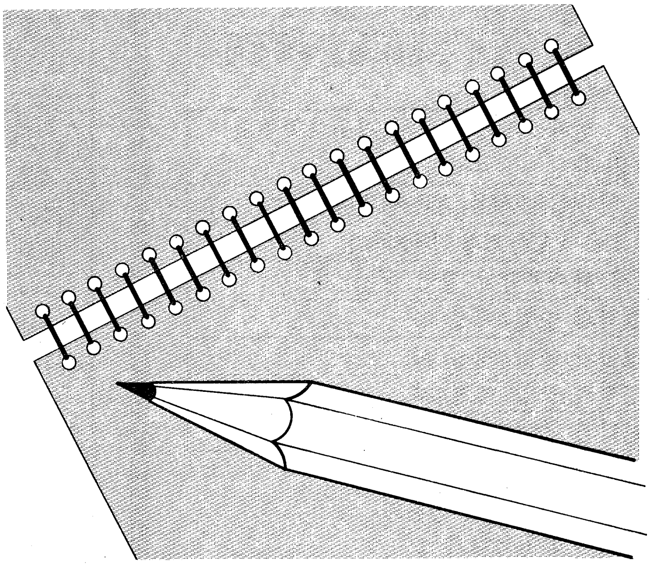
취급품목
- IBM COMPATIBLE PC 386/AT/XT
- ROM DISK DRIVE
- TRANSPUTER
- BSC CARD
- SNA 3270
- 기타 PC 주변기기

* 주소 : 대구시 달서구 부유3동 493-7
TEL: (053) 60-3700

DIGITAL TECHNOGY

주소 : 서울시 서초구 방배1동 983-41호 (구미빌딩)
TEL: (02) 581-1511 (대표)
583-5593 (A/S부)
FAX: (02) 588-2791

랩탑 컴퓨터



모든 것이 소형화·경량화·고급화되는 추세의 현대사회에서 컴퓨터도 예외는 아니다. 초창기의 컴퓨터는 웬만한 건물 한 채 정도의 부피와 중량을 가지고 있어 지금의 데스크 탑형 컴퓨터와는 비교조차할 수 없이 웅장(?)하였다. 하지만 초창기 컴퓨터의 성능은 큰 덩치에 비하여 지금의 데스크 탑형 컴퓨터 성능과는 비교가 될 수 없을 정도로 기능과 처리능력이 매우 빈약하였다.

그러나 꾸준한 기술개발과 크기의 소형화로 성능은 더욱 향상되고 크기는 더욱 작아진 컴퓨터가 등장하게 되었고, 랩탑(laptop) 컴퓨터도 이러한 배경으로 탄생하게 되었다.

랩탑컴퓨터는 기존의 데스크 탑형 컴퓨터가 가지고 있는 기능과 거의 유사한 기능과 성능을 가지고 있거나 더 나은 성능을 가지면서도, 크기가 작고 배터리 전원을 사용할 수 있어 전원이 없는 곳에서도 사용할 수 있고, 사람이 갈 수 있는 곳이라면 어느 곳이라도 이동할 수 있는 이동의 편리성을 특징으로 하는 컴퓨터이다.

구성

랩탑컴퓨터는 CPU를 8088, 80286, 80386중에 어느 것을 채택하였나에 따라 그 종류가 구별된다. 각각의 종류에 따른 개략적인 구성은 그림 1, 2, 3과 같으므로 참조하기 바란다.

변천과정

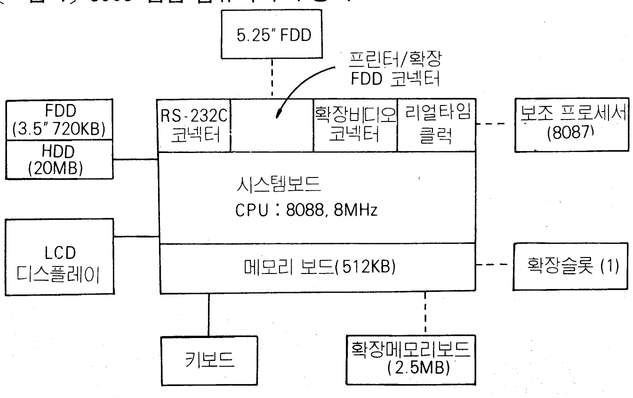
랩탑의 변천과정은 우선 모니터의 변화에 따라 구분할 수 있다. 초기의 랩탑컴퓨터는 LCD(Liquid Cristal Display)라는 액정 모니터를 사용하였다.

이것의 원리는 유리 사이에 액체 수정을 담은 셀로서 이루어졌고, 유리판 안쪽의 투명한 전극판에 전압이 걸리면 그 사이에 있는 액정의 분자 운동 방향이 달라져서 이것을 통과하는 빛의 변화에 따라 숫자나 문자를 표시한다. 이것은 스스로 발광하지 않기 때문에 전력 소비가 적고 배터리를 사용할 수 있다. 가격이 싸다는 장점이 있지만 반면에 반사각도에 따라 화면이 흐려지고, 반응속도가 느리며, 어두운 장소에서는 글자를 판독하기가 어렵다는 단점이 있다.

LCD모니터 이후 가스 플라즈마(GAS PLASMA) 모니터가 개발되었다. 이것은 여러장의 유리를 붙여 기체를 넣어 방전시키는 원리를 이용한 것으로 화면이 상당히 밝고, 반응속도가 빨라 상당히 기대를 모았다. 그러나 이 제품은 전력 소모량이 대단히 많아 배터리를 사용하기 힘들다는 것이 최대의 약점이 되고 있다.

이후로 개발된 것이 배후조명 LCD(backlight LCD)이다. 이 배후조명 LCD는 기존의 LCD셀 뒤쪽에 따로 조명장치를 부착시킨 것이라고 볼 수 있다. 배후조명에 따라 EL 백라이트, CCFT

〔그림 1〕 8088 랩탑 컴퓨터의 구성예



형태로 구분할 수 있다.

EL 백라이트 LCD는 종전의 LCD가 어두운 곳에서 판독하기 어려운 점에 착안하여 거의 플라즈마의 밝기와 유사한 수준으로 끌어 올림으로써 이 문제를 해결하였다. 이와같은 백라이트 LCD는 현재 랩탑시장에서 LCD 화면의 표준으로 되고 있다.

또한 1988년 중반부터 사용되기 시작한 CCFT백라이트 LCD가 있는데 이것은 Cold Cathode Fluoresce Tube 즉 냉음극 형광등을 사용하여 화면이 더욱 선명해진 것이 특징이다.

마지막으로 EL모니터 방식이 있는데 이것은 ELECTRO LUMINESCENCE (전계발광)의 약자로 투명전극을 붙인 유리판에 형광물질을 수미크론 두께로 하고 뒤에는 빛이 반사하도록 금속을 진공증착하여 발광시키는 방법이다. 이 방법을 이용하면 가장 높은 해상도를 얻을 수 있다.

구입시 주의 사항

첫째 자신의 사용 용도에 알맞는 제품을 고른다.

랩탑컴퓨터는 모델에 따라 특징이 뚜렷하므로 자신이 사용하고자 하는 목적에 알맞게 선택해야 한다. 즉 단시간내에 선명한 화면상태를 요하는 사람은 가스플라즈마나 EL타입의 랩탑을 선택하고, 오랜 시간을 이용하고자 하는 사용자는 전력을 적게 소비하는 LCD형식의 랩탑을 선택하는 것이 현명하다.

둘째 휴대가 용이한 제품을 택한다.

랩탑컴퓨터의 가장 중요한 특징중의 하나가 즉시 어느 장소로든지 이동이 가능하다는 것이다. 이를 위해서는 배터리를 사용할 수 있어야 한다. 또한 운반하기 쉽게 크기가 작고, 무게가 가벼워야 한다. 여기서 집고 넘어가야 할 사항은 사양서에 표시된 랩탑의 무게가 배터리, 전원부 등의 모든 주변장치가 포함된 무게인지, 이것이 제거된 무게인지 정확하게 알아보아야 한다.

한가지 덧붙여 말하면 자동차 배터리에 전원을 연결하여 쓸 수 있는지를 살펴보아야 한다. 기동성을 생명으로 하는 랩탑컴퓨터는 언제 어느 곳에서든지 전원 공급에 차질이 없어야 한다. 전원없이는 아무리 훌륭한 성능을 갖춘 컴퓨터라도 아무런 쓸모없는 쇳덩어리에 지나지 않기 때문이다.

셋째 어떠한 기능이 지원되는가?

랩탑컴퓨터는 리모트 콘트롤 기능을 중시하므로 모뎀을 장착할 수 있는지, FAX카드의 설치가 가능한가, 기존의 데스크탑형 컴퓨터와 연결할 수 있는지를 세밀하게 살펴 보아야 한다. 한가지 예로 5.25인치 플로피 디스크 콘트롤러 코넥터가 표준 규격인지 그 회사의 독자적 규격인지를 살펴 보아야 한다. 해상도도 랩탑컴퓨터의 가격을 좌우하는 중요한 요소이므로 CGA전용인지, EGA용인지, 허큘리스용인지, VGA용인지를 살펴두어야 한다. 또한 구입한 제품을 앞으로도 계속 사용할 수 있는지 등의 장래성을 고려해야 한다.

네째 보수가 용이한가?

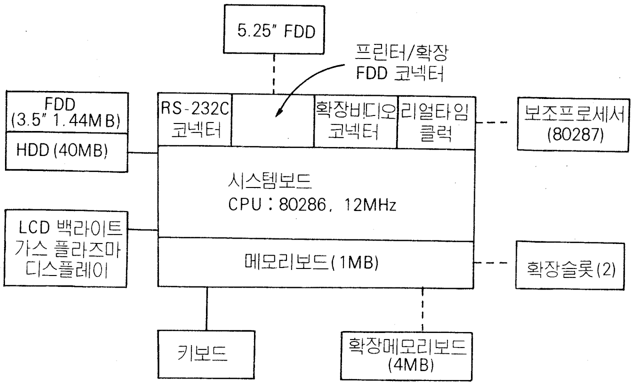
랩탑컴퓨터는 기존의 데스크탑형 컴퓨터와는 달리 이동성이 강조되므로 그만큼 파손의 위험에 노출이 되어 있다. 실제로 플라스틱으로 되어 있는 케이스와 모니터의 파손율이 높다. 랩탑컴퓨터는 기존 데스크탑 컴퓨터의 규격과는 많은 차이가 있어 쉽게 시중에서 부품을 구하기가 어려우며, 간단한 수리 조차도 하기 힘든 여건이다. 그러므로 철저하게 사후 보장을 받을 수 있는 제품을 선택해야 한다.

다섯째 한글 사용이 가능한지?

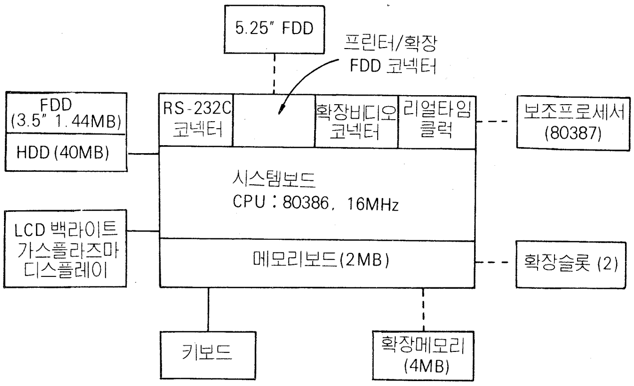
한글이 지원되려면 해상도가 600x400 정도가 되어야 한다. 600x200 의 해상도는 영어권에서는 문제가 되지 않지만 가로 40자 세로 25자의 한글을 표시하려면 한글 한 자에 가로, 세로 16픽셀 정도가 되어야만 글씨를 알아보기도 쉽고 글자 모양도 예쁘게 되기 때문이다. 랩탑컴퓨터는 거의 수입제품이고 이들은 EGA(640×350나 VGA(640×480, 320x200)와 같은 고해상 컬러 디스플레이를 지원하지만 하드웨어적인 한글 처리가 되지 않는다.

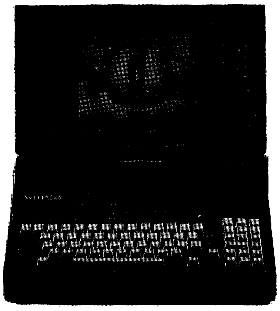
그러므로 국내에서 널리 사용되는 한글 소프트웨어의 지원이 어렵다. 물론 한글이 지원되는 랩탑컴퓨터도 있지만, 거의 대부분의 한글은 소프트웨어적 처리로 되어 있어 처리 속도가 느리고 기존의 한글 응용소프트웨어의 지원이 원활하지 못하다. 이 한글지원 문제가 랩탑컴퓨터의 보급에 상당한 저해 요인으로 작용하고 있다. PC

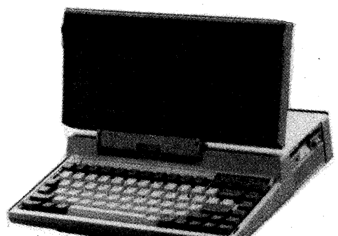
〔그림 2〕 80286 랩탑 컴퓨터의 구성예



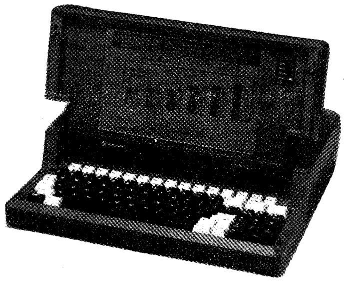
〔그림 3〕 80386 랩탑 컴퓨터의 구성예

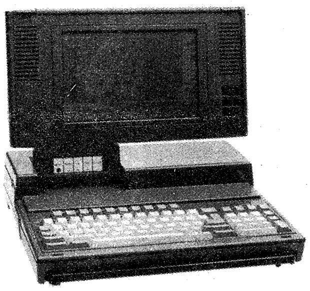
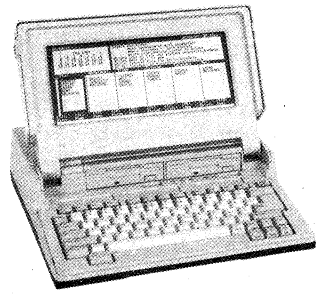


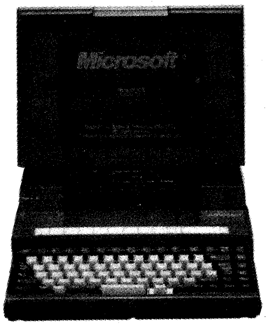
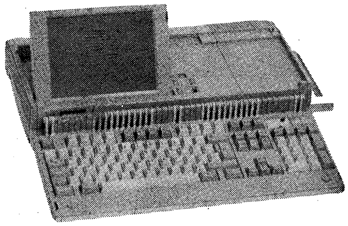
제품 / 규격			
모델명	LP-3300	PRO-8000	PRO-8100
생산업체	모던 컴퓨터	대우통신	대우통신
공급업체	뉴텍 코리아 (02-585-3891)	대우통신 (02-771-0035)	대우통신 (02-771-0035)
프로세서	80286	80C86-10	86C286-12
메모리	1MB램(4MB 확장 가능)	512KB(2.5MB 확장 가능)	512KB(2.5MB 확장 가능)
처리속도	6MHz/12MHz	8MHz/10MHz	8MHz/12MHz
디스플레이 종류	LCD	LCD 백라이트	LCD 백라이트
디스플레이 해상도	640×400	640×400	640×400
디스크드라이브 FDD	3.5″ 1.44MB×1대	3.5″ 720KB×2대	3.5″ 1.44MB×1대
디스크드라이브 HDD	20MB	—	20MB
키보드	95키	81키	81키
확장슬롯	2	1	1
표준 인터페이스	• EGA/MGA/CGA 모니터 포트 • RS-232C 시리얼 포트 • 외부 5.25″ 1.2MB FDD 포트 • 병렬 프린터 포트	• CGA/MDA/허클리스 • RS-232C 시리얼 포트 • 마우스 포트 • 센트로닉스 병렬 포트 • 외부 비디오 포트 • 외부 FDD 포트	• EGA/CGA/MDA/허클리스 • RS-232C 시리얼 포트 • 마우스 포트 • 센트로닉스 병렬 포트 • 외부 비디오 포트 • 외부 FDD 포트
전원 외부전원	가능	가능	가능
전원 배터리	가능	가능	가능
모뎀	내장 가능	내장 가능	내장 가능
크기(cm), 무게(kg)	38×34×9, 8	36.2×33×8, 5.8	36.2×33×8, 5.9
한글처리	가능	SW적으로 가능	SW적으로 가능
가격(원)	3,630,000	?	?
기타	• 외부 5.25″ 1.2MB FDD • 1200/2400 모뎀 • MS-DOS 3.2 • EMS 드라이버 프로그램	• 300/1200/2400 모뎀 • 외부 IBM호환 컬러, 모노 모니터 • 외부 자동차 어댑터 • MS-DOS 3.×× • GW-BASIC 3.××	• 300/1200/2400 모뎀 • 외부 IBM EGA/컬러, 모노크롬 모니터 • 외부 자동차 어댑터 • MS-DOS 3.×× • GW-BASIC 3.××

제품 / 규격				
모델명		PRO-8200	T1200	T3200
생산업체		대우통신	도시바	도시바
공급업체		대우통신(02-771-0035)	창명실업 (02-757-1049)	창명실업 (02-757-1049)
프로세서		80386SX-16	80C86	80286
메모리		512KB(2MB 확장 가능)	1MB	1MB
처리속도		8MHz/16MHz	9.54MHz	6MHz/12MHz
디스플레이	종류	LCD 백라이트	supertwist LCD	가스 플라즈마
	해상도	640×400	640×200	640×400
디스크드라이브	FDD	3.5″ 1.44MB×1대	3.5″ 720KB×1대	3.5″ 720KB×1대
	HDD	20MB	20MB	40MB
키보드		81키	82키	85키
확장슬롯		1	1	2
표준 인터페이스		• VGA/EGA/CGA/MDA/허큘리스 • RS-232C 시리얼 포트 • 마우스 포트	• RS-232C 시리얼 포트 • 외부 5.25″ FDD 포트 • RGB 컬러 모니터 포트 • 모노크롬 콤포지트 모니터 포트 • 숫자 키패드 포트	• RS-232C 시리얼 포트 • EGA 컬러 모니터 포트 • 5.25″ FDD 포트 • 선택적 병렬 프린터 포트
전원	외부전원	가능	가능	가능
	배터리	가능	가능	불가능
모뎀		내장 가능	내장 가능	내장 가능
크기(㎝), 무게(㎏)		36.2×33×8, 5.9	31×30×67, 5	40×37×10, 8.5
한글처리		SW적으로 가능	불가	불가능
가격(원)		?	4,020,000	6,520,000
기타		• 300/1200/2400 모뎀 • 외부 IBM VGA/EGA 컬러, 모노크롬 모니터 • 외부 자동차 라이트 어댑터 • MS-DOS 3.×× • GW-BASIC 3.××	• 300/1200 모뎀 • 숫자 키패드 • 1MB EMS카드 • MS-DOS 3.2	• 외부 5.25″ FDD • 내부 3MB 메모리 카드 • MS-DOS 3.2

●구입가이드●

제품 / 규격			
모 델 명	T5100	DG ONEMOEL 2T	S5000
생 산 업 체	도시바	데이터 제너럴	삼성전자
공 급 업 체	창명실업(02-757-1049)	정원 시스템(02-542-3675)	삼성전자(02-559-2652~4)
프 로 세 서	80386	80C88-2	80286
메 모 리	2MB(4MB 확장 가능)	512KB(2.5MB 확장 가능)	1MB
처 리 속 도	8MHz/16MHz	4.77MHz/7.16MHz	10MHz
디스플레이 종 류	가스 플라즈마	LCD	supertwist LCD
디스플레이 해상도	640×400	640×256	640×400
디스크드라이브 FDD	3.5″ 1.44MB×1대	3.5″ 720KB×1대	3.5″ 1.44MB×1대
디스크드라이브 HDD	40MB	20MB	20MB
키 보 드	82키	80키	85키
확 장 슬 롯	2	2	—
표준 인터페이스	•모니터 포트 •내장 2MB 메모리 카드 •선택적 병렬 프린터 포트 •5.25″ FDD 포트 •RS-232C 시리얼 포트	•병렬 프린터 포트 •RS-232C/RS-422 포트	•센트로닉스 호환 병렬 포트 •RS-232C 시리얼 포트 •RGB 외부 모니터
전원 외부전원	가능	가능	가능
전원 배 터 리	불가능	가능	?
모 뎀	내장 가능	내장	내장 가능
크기(cm), 무게(kg)	36×31×8.9, 6.8	37×30×8, 4.8	35.6×33×7.7, 5
한 글 처 리	불가능	불가능	SW적으로 가능
가 격(원)	8,410,000	2,500,000	?
기 타	•외부 5.25″ FDD •내부 2MB 메모리 카드 •1200 모뎀 •MS-DOS 3.2	•1200/1300 모뎀 •3.5″ FDD •MS-DOS 3.2	•인텔 80287 보조프로세서 •MS-DOS 3.3

제품 / 규격				
모델명		TURBO LAPTOP 3400	LT-3400	Tandy 1400LT
생산업체		치코니	치코니	래디오 섁
공급업체		글방컴퓨터(02-588-1747)	한국 스카다 시스템(576-0322)	북두 컴퓨터(02-785-0864)
프로세서		80C286	80286-16	NEC V-20
메모리		1MB	1MB	786KB
처리속도		8MHz/16MHz	—	4.77MHz/17.16MHz
디스플레이	종류	가스 플라즈마	가스 플라즈마	LCD
	해상도	640×400	640×400	640×200
디스크드라이브	FDD	3.5" 1.44MB×1대	3.5" 1.44MB×1대	3.5" 720KB×2대
	HDD	40MB	40MB	20MB
키보드		85키	84키	76키
확장슬롯		1	1	1
표준 인터페이스		• RS-232C 시리얼 포트 • 병렬 포트	• 외부 5.25" FDD 포트 • 병렬 프린터 포트 • CGA/MGA/EGA 모니터 포트 • RS-232 시리얼 포트	• RS-232C 시리얼 포트 • 병렬 프린터 포트
전원	외부전원	가능	가능	가능
	배터리	가능	불가능	가능
모뎀		내장 가능	내장 가능	내장 가능
크기(㎝), 무게(㎏)		35×32×10, 5.5	37.5×33.6×9, 7	31.8×36.8×8.9, 6
한글처리		그래픽 한글처리	불가능	그래픽 한글처리
가격(원)		3,500,000	3,500,000	2,300,000
기타		• 외부 5.25" FDD • 인텔 80827 보조프로세서 • 1200bps 모뎀 • 한글 DOS	• 외부 5.25" FDD • 외부 키보드용 폰잭	• MS-DOS 3.2

제품 / 규격				
모 델 명		LIB/LAPTOP	HALIKANLA-30A	PPC640
생 산 업 체		립텍	CHAPLEP	AMSTRAD
공 급 업 체		립텍코리아 (02－702－0662)	메이트컴퓨터 (02－272－9484)	해태상사(02－717－0100)
프 로 세 서		80286-12	80286-12	8086
메 모 리		1MB	1M	640KB
처 리 속 도		12MHz/16MHz	12MHz	8MHz
디 스 플 레 이	종 류	가스 플라즈마	LCD	LCD
	해상도	640×400	640×400	640×200
디스크드라이브	F D D	3.5″ 720KB×1대	3.5″ 1.44MB×1대	3.5″ 720KB×2대
	H D D	—	20MB	—
키 보 드		98키	81	102
확 장 슬 롯		1	2	—
표준 인터페이스		• RS-232C 시리얼 포트 • 병렬 프린터 포트	• RS-232C 포트 • 병렬 프린터 포트	• RS-232C 포트 • 병렬 프린터 포트
전 원	외부전원	가능	가능	가능
	배 터 리	가능	가능	가능
모 뎀		내장	내장	내장
크기(㎝), 무게(㎏)		37.5×33.6×9, 7	40.8×32.4×8, 6	45×23×10, 5.8
한 글 처 리		그래픽 한글처리	SW적으로 가능	SW적으로 가능
가 격(원)		3,850,000	2,980,000	1,250,000
기 타			• 외부 모니터 코넥터 • MS-DOS 3.3 • GW-BASIC	• MS-DOS 3.3

월간 퍼스널컴퓨터의 정기구독자에게는 아래와 같은 특전이 마련되어 있습니다.

—정기구독자는 본사가 보유한 각종 정보 및 자료를 무료로 배부받으실 수 있읍니다.
—본지에 게재되는 각종 프로그램을 디스켓에 담아 무료로 보내드립니다.
—본지 및 본지의 자매지(월간 전자과학) 주최로 개최되는 세미나 및 각종 행사에 무료로 참석하실 수 있읍니다.
—본지에서 행하는 각종 부대사업에 할인혜택이 주어집니다.
—구독기간중 책 가격이 인상되더라도 종전가격으로 구독이 가능합니다.

● 구독신청은

—시중은행에서 지로용지를 이용하시거나 우체국에서 대체구좌 또는 소액환으로 송금하시면 됩니다.
은행 지로번호 : 3000667
우체국 대체구좌번호 : 010256—31—0526772

● 정기구독료는

—1 년 구독시 30,000원
—6개월 구독시 15,000원

● 구독신청시 주의사항은

—구독기간과 주소(우편번호), 전화번호와 받으실 분의 성명을 정확히 기입하시기 바랍니다.
—배달사고나 기타 변동사항은 본지 영업부로 즉시 연락하시기 바랍니다. 신고전의 분실사고는 책임지지 않습니다.

● 정기구독 문의처는

—서울시 용산구 갈월동 17 ㊥ 140-150
월간 퍼스널컴퓨터 영업부
전 화 : (02) 754—1044(대표)
FAX : 757—1049

▨ 전국총판 ▨

●서울●	종로, 중구, 마포, 용산	명 문 사	774—6314
	중랑, 성동, 동대문	합 동 사	242—8772
	노원, 성북, 도봉	진 원 사	996—2443
	강남, 강동, 송파	광 일 사	415—7330
	관악, 동작	관 악 사	889—4504
	서대문, 은평	형 제 사	304—9462
	영등포, 구로	영 도 사	613—5381
	강서, 양천	연 우 사	865—8239
●부산●		종합기술서적	462—2431
		영광도서전시관	805—9500
		북 부 총 판	805—9677
		남 부 총 판	756—2727
●대구●		청 운 서 림	23—7335
		한 일 서 적	23—7101
		학 원 사	22—0336
●인천●		동 양 서 점	73—3384
●광주●		삼 복 서 점	22—0258
●경기●	부천	대 경 서 점	62—3672
	안양, 안산	사 랑 글 방	57—1916
	수원	동 아 서 점	5—6436
	성남	영 광 서 점	2—6532
●강원●	원주	원주종로서적	42—2943
	삼척	영 동 사	72—6060
	강릉	삼 일 사	2—2314
	춘천	대 진 서 점	54—5148
●충북●	충주	문 학 사	2—2814
	청주	화 문 당	55—0503
●충남●	천안	동방서적공사	63—3314
	대전	동 양 서 점	22—1538
●경북●	포항	경 북 서 점	2—2717
	구미	중 앙 서 점	52—7915
●경남●	진주	동 아 서 점	42—3768
	진해	학 애 서 점	2—2523
	울산	우 정 서 점	43—6003
	마산	기 술 서 적	56—1417
●전북●	전주	홍 지 서 점	6—5311
	군산	동 원 서 점	42—3776
●전남●	목포	문 화 당	5—3086
	여수	대 양 서 점	62—2111
●제주●		광 양 서 적	53—5077



□ 프로그램 제공안내 □

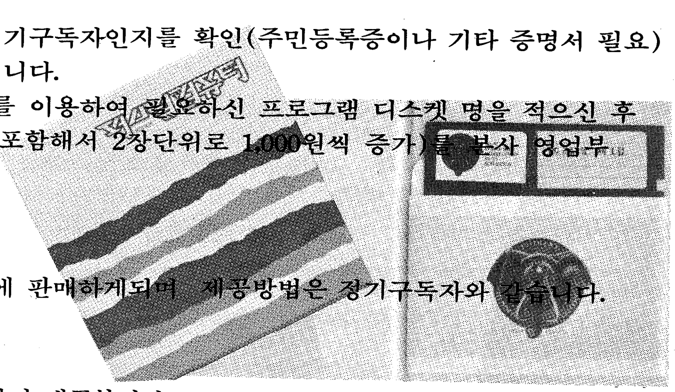
컴퓨터의 보급이 늘어남에 따라 이제는 어디에서나 쉽게 접할 수 있을만큼 대중화되었으며 컴퓨터란 용어자체를 자연스럽게 대화속에 끌어들일 수 있게 되었습니다. 그러나 아직까지는 활용할 수 있는 소프트웨어가 부족하다고 할 수 있습니다. 저희 퍼스널컴퓨터에서는 이러한 분들을 위하여 매달 책에 게재되는 프로그램을 디스켓에 담아 제공하고자 합니다.

□ 프로그램 제공방법

●정기 구독자의 경우

—직접 찾아오시면 본사 영업부 디스켓 담당직원이 정기구독자인지를 확인(주민등록증이나 기타 증명서 필요) 한후 프로그램 디스켓 1장당 1,500원씩에 판매하게 됩니다.

—지방에 계신 분은 본지 뒷면에 있는 은행지로 용지를 이용하여 필요하신 프로그램 디스켓 명을 적으신 후 디스켓 구입비(1장당 1,500원씩 증가)와 송료(자켓 포함해서 2장단위로 1,000원씩 증가)를 본사 영업부 프로그램 디스켓 담당자 앞으로 보내주시면 됩니다.



●일반 독자의 경우

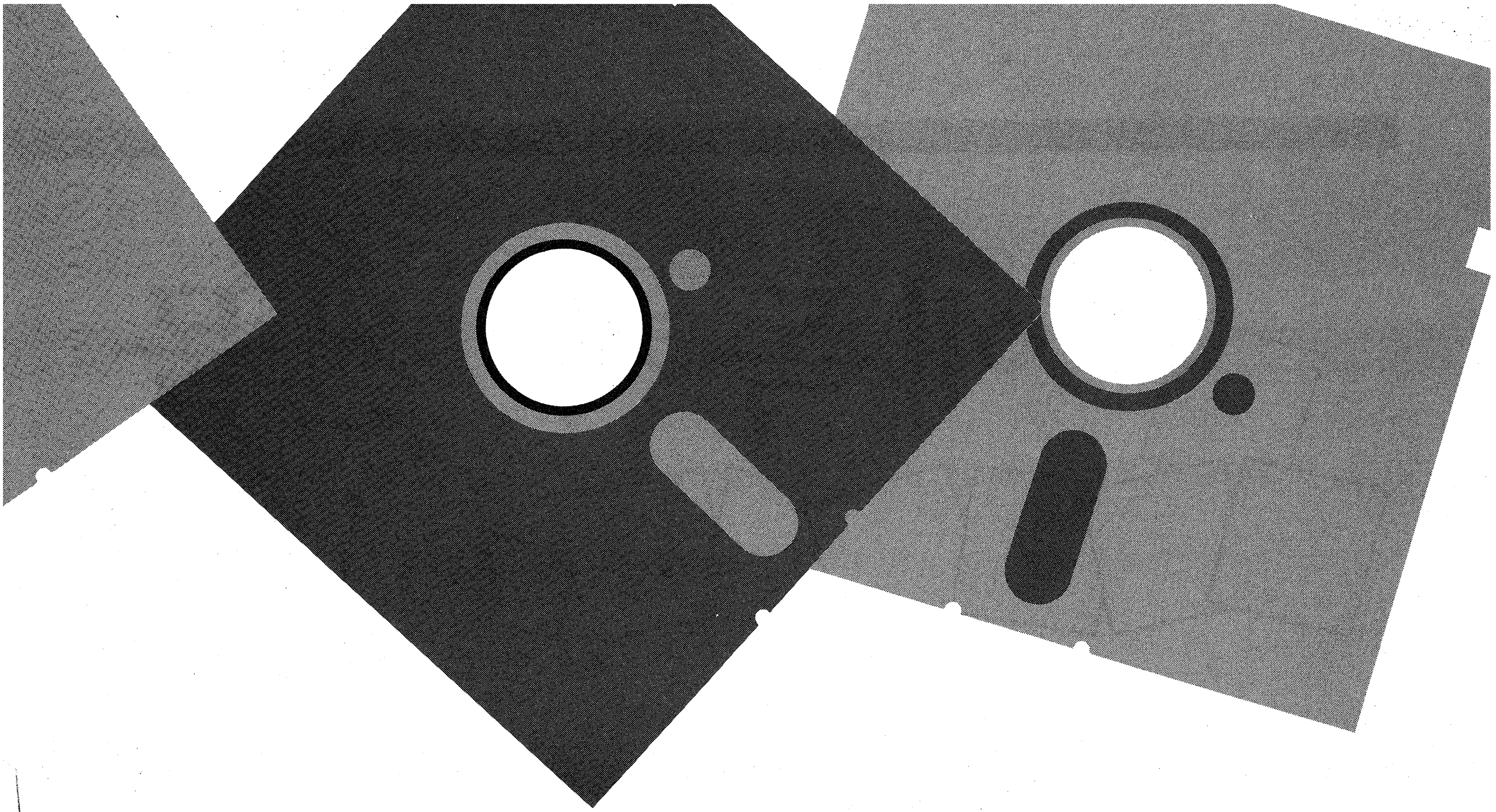
—일반 독자의 경우 프로그램 디스켓 1장당 4,000원씩에 판매하게되며 제공방법은 정기구독자와 같습니다.

□ 알림

독자 여러분의 요청에 따라 월별로도 프로그램을 수록하여 제공합니다.

□ 문의처

주소 : 서울시 용산구 갈월동 17 세기빌딩 3층 월간 퍼스널컴퓨터 영업부 디스켓 담당자앞
전화 : (02)754-1044(대표) FAX : (02)757-1049



프로그램 디스켓

▨ 제1집

디스크 논리구조를 이용한 화일복구(88년 12월호 p74)
플로피디스크상에서 실수로 지워진 화일들을 완벽하게 복구시켜 준다.

LX-800의 버그(89년 3월호 p160)
LX-800에서 그래픽을 하드카피할때 생기는 버그를 제거해준다.

스피디 3의 제작(89년 2월호 p166, 3월호 p192)
마이크로 로보트의 소스 프로그램

360KB이상의 데이터를 복사하려면(89년 4월호 p54)
360KB 이상의 데이터를 분리하여 저장할 수 있게 해준다.

레코드판 관리 프로그램(89년 4월호 p94)
200개의 곡명을 수록할 수 있으며 원하는 형태별로 찾아볼 수도 있다.

미분이 가능한 계산기 프로그램(89년 4월호 p110)
공학에서 자주 사용되는 복잡한 계산을 처리할 뿐만 아니라 미분도 가능하다.

그래픽 애니메이터(89년 4월호 p176)
20개의 프레임에 연속적인 이미지를 그린 다음 빠른 속도로 출력시켜 움직임을 표현한다.

QUICKEY.COM(89년 4월호 p200)
램상주 유틸리티로서 키보드의 감도를 빠르게 해준다.

교육용 프로그램(89년 5월호 p76)
물리학습용 프로그램으로 교육용 프로그램의 구성은 어떻게 되어 있나를 알 수 있다.

전화번호부 관리 프로그램(89년 5월호 p88)
디베이스로 간단히 작성한 프로그램으로 전화번호와 주소를 입력하여 검색하거나 우편용 레이블 등을 작성할 수 있다.

□ 주의
각 화일들은 아카이브화 되어 있기 때문에 확장시킬 때 용량이 모자랄 수도 있으므로 이점에 유의하여야 하며 프로그램을 실행시키기 전에 반드시 README 화일을 불러 확인한다.

유틸리티

▨ 마우스 활용 툴

마우스 드라이버 툴키트(89년 5월호 p94)
마우스 드라이버 프로그램으로 각자가 손쉽게 자기용도에 맞게 응용할 수 있는 프로그램 툴이다.

그래픽 에디터 SEM-DRAW(89년 5월호 p112)
여러가지 그래픽에 필요한 루틴들을 첨가하여 마우스를 이용해 손쉽게 그래픽을 처리할 수 있도록 한 에디터이다.

SEM-DRAW에 이미지 저장기능을(89년 7월호 p66)
이미지 저장기능과 여러가지 다양한 폰트를 사용할 수 있게 하였다.

▨ BBUN(89년 6월호 p100)
터보 에디터의 장점을 모두 갖춘 새로운 에디터

▨ 취업정보안내 프로그램(89년 6월호p214)
퍼스널 컴퓨터를 이용하여 취업정보안내에 관한 각종 자료들을 데이터베이스화하여 작성한 프로그램

▨ 윈도우 라이브러리

NKP에서의 윈도우 구현(89년 6월호 p90)
NKP상에서 윈도우를 구현해주는 윈도우 라이브러리로서 프로그램 작성시 다양하게 이용할 수 있다.

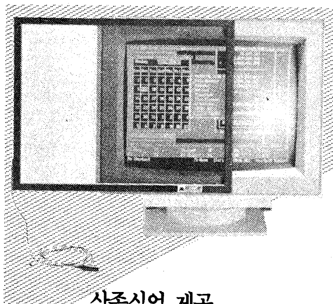
NKP용 도스 유틸리티(89년 7월호 p72)
팝업메뉴 방식의 도스 유틸리티로서 각종 코맨드가 한글로 처리되어 있다.

공개 소프트웨어

▨ DX.COM(89년 5월호 p198)
디렉토리 관리 기능을 가진 DX.COM 등 여러가지 다양한 프로그램들이 수록되어 있다.

▨ 와일드 캣(89년 7월호 p.124)
BBS용 호스트 소프트웨어로서 가장 많이 사용되는 프로그램중 하나이다.

애독자 선물



3명

폴리에스터에 특수 구리도금과
아크릴 수지로 가공 처리된
첨단기술의 V.D.T 증후군
방지용 컴퓨터 보안기이다.

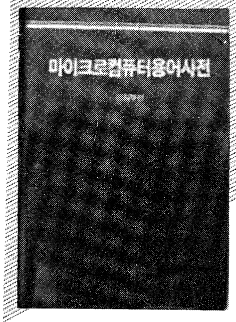
삼종실업 제공
TEL : 784-6931



5명

컴퓨터에 전문지식이 없는 사람도
쉽게 접근할 수 있도록
꾸며진 좋은 지침서이다.

영진출판사 제공
TEL : 719-1119



5명

누구나가 한권씩은 지니고 있어야
하는 컴퓨터 용어사전으로 비교적
자세히 설명되어 있다.

오므사 제공
TEL : 754-1044



1명

A-400 마우스는 IBM PC/XT/
AT 및 호환기종이면 기종에
관계없이 사용이 가능하다.

하나컴퓨터 제공
TEL : 717-4894



3명

특수 산화물 코딩 처리된
에러율 0%의 디스켓이다.

한국쓰리엠 제공
TEL : 785-4567



5명

미국 UL규격을 인정받은 제품으로
아크릴 시트로 특수가공되어
선명한 화면을 유지시켜준다.

화신전자 제공
TEL : 733-3567

대잔치



5명

SKC 제공
TEL : 756-5151

균일한 자성체의 배양과 성장으로 도막의 균일도 및 기록신호의 안정도를 극대화하였다.



1명

(주)유원 마케팅 제공
TEL : 273-7729

커서의 정확한 조정을 위한 콘트롤 레버와 왼손잡이, 오른손잡이 모두 사용할 수 있는 제3 발사버튼이 부착되어 있는 것이 특징이다.

※ 월간 퍼스널컴퓨터는 독자 여러분의 의견을 소중히 하여 책의 편집에 반영하고자 합니다.
애독자 설문지를 사용하여 질의내용에 성의껏 답하시어 15일까지 보내주시면 28명을 뽑아 선물을 드립니다.

※ 서울 거주자는 8월31일까지 신분증을 지참하시고 직접 본사로 나오시면 해당 상품을 드리며, 지방에 계신 분은 우송해 드리겠습니다.
기간은 8월31일까지이며, 이후는 무효로 처리합니다.

※ 애독자 설문지는 249페이지에 있습니다.

애독자 선물대잔치
7월호 당첨자 명단

● 컴퓨터 보안기

이 동희 충북 청원군 남일면 고은 3구 238번지
강 선모 대전시 유성구 대덕단지 우체국 사서함 3호 레이저실
이 지헌 부산시 동구 수정 5동 471-40번지 1통 2반

● MS-DOS 입문

최 영숙 서울시 서대문구 홍제 4동 19-5
이 웅영 경기도 양평군 단월면 산음 2리 283번지
김 대영 경남 진주시 대안동 15-1번지
김 종호 강원도 삼척시 남양동 22-13 8통 3반
황 성규 서울시 관악구 봉천 6동 100-16 24통 2반

● 디스켓 박스(10장들이)

박 일수 서울시 용산구 용산동 2가 1-362
김 태용 인천시 북구 구산동 34-1
최 완주 부산시 동구 수정 3동 1163-13번지 15통 6반
노 진우 강원도 속초시 교동 주공아파트 9동 102호
서 광석 충남 천원군 성환읍 성월리 144 성환 주택가동 102호

● 마우스

박 영호 경기도 광명시 광명 3동 129-1호

● 디스켓 자켓(4장들이)

박 중규 충북 청원군 강내면 월탄리 301
박 광덕 부산시 남구 남천동 63 KBS부산방송본부 라디오 기술부
신 병원 경기도 용인군 기흥읍 구갈리 216

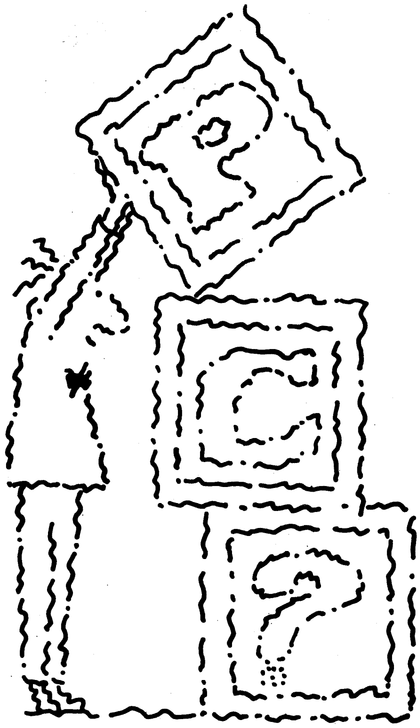
● 컴퓨터 용어사전

송 준 서울시 강남구 청담동 효성빌라 3-104
김 광형 경남 창원시 용호동 72번지 용지아파트 320동 503호
이 순미 전남 순천시 매곡동 43-6 7통 3반
정 석주 경기도 부천시 도당동 222-1 (주)아남산업 자재과
이 윤순 대구시 동구 율하동 943-5

● 화신컴퓨터 보안기

김 익수 경북 경주시 서악동 1131-1
전 재홍 서울시 강남구 대치 4동 동아아파트 다동 505호
이 승례 충북 청원군 문의면 미천리 27 121-52번지
현 철 서울시 은평구 역촌동 1-12 성가 가축병원
고 철성 서울시 마포구 공덕 1동 111-142 17통 5반

책을 마감하고



●아흡!

끼야~ 호!
내일이면
제주로 간다. 〈榮〉

●무척 덥습니다.

시원한 바닷물에 풍덩 들어가 헤엄이라도 쳐야겠습니다.

반짝이는 별을 보며 정다운 이야기도 나누어야 겠습니다.

이번 여름은 색다른 추억으로 간직될 것입니다.

지금의 평온과 여유를 느낄 수 있게 도와주신 여러분들께 감사 드립니다. 〈E. Y〉

●우리들이 갖고 있는 천연자원 중에서 최대의 낭비는 자기들의 무한한 가능성을 개발하지 않는 무수한 사람들 입니다.… 당신이 할 수 없다고 생각하시면 당신은 하지 못합니다. 그러나 할 수 있다고 생각하면 당신은 할 수 있는 기회를 쉽게 포착하실 수 있지요. 해보려고 노력하는 것만으로도 당신은 새로운 인간이 된듯 한 기분이 될 것입니다. 명성이란 될 것 같지 않은 일을 찾아내서 해볼 때 생기는 것이지요. —신해진역 카피 카피 카피중에서— 〈김현〉

●두번째!

장마가시작됐다고하던데그래서이렇게푹푹찌는걸까여름은역시더워야한다고누군가그러던데이러다혹시퍼스널컴퓨터보면서조는것은아닌지금새한달이지나갔나보다편집후기달라고하는거보니까엉망진창이라는말이있지장마철이라서하는소리가아니고그만큼또실수를했나보다이책을보고있는독자들은무슨소리인지모르겠지.

이제,

장마가지나후덥다고혹퍼스널컴퓨터보면서졸기없기재미없다고책꽂이비었다고꽂아놓기없기그리고내용좋다고혼자보기없기더욱주의해야할것은혼자만알고있지말고다같이알수있도록지금보고있는이책에소개하자는것지면관계상빈칸이없음한치의땅아니지면을놀리지않기위한것임. 〈榮珠〉

●한달동안 긴장의 연속이었다.

원고를 제출할 때마다,
아직도 멀었나보다.
깨어질 때마다 하나씩 터득해 나가고 있지만.
앞으론 깨어지더라도 좀 더 높은(?) 차원에서 깨어져야 겠다. 〈圭〉

●빨리 먹는 밥에 체한다는 말이 있습니다. 며칠전 과학기술처에서 수퍼컴퓨터 개발 소동으로 사람들을 놀래켰던(?) 일이 있었습니다.

사람들의 욕심들은 한이 없다고 합니다. 그러나 우리에게 있어 가장 중요한 것은 허구에 대한 믿음이 아니고 진실에 대한 갈망입니다. 〈둘리〉

●뒤죽박죽 엉킬대로 엉킨 한달이었다.

더운 날씨탓으로 돌린다면 너무 무책임한 말이겠지.

연휴와 휴가로 말미암아 짧은 기간에 책을 펴내야 했고, 그로 인해 파생되었던 작은 이해와 갈등들…

하지만 자기 주장·권리 못지 않게 자기 의무 또한 중요하다는 생각, 서로 이해하고 아끼는 마음으로 임했다면 좀더 시원한 여름을 보내지 않았을까 하는 아쉬움을 남긴채 휴가를 떠나련다. 〈七〉

●좋다.

과장, 치장없이
건네는 소박한 말 한마디
고맙다.
그 한마디로 큰 위안이 되기에
행복하다.
내 주위에 자그마한 것을 소중히
여기고 베푸는 이들이 있기에
소중한 그들에게 전하고픈 말
고마워요! 〈賢〉

●골인점에 도달한 마라톤 선수처럼 피곤과 공허감에 숨쉴틈 없는 마감속에 바싹 다가온 휴가철로 바쁜 때에도 들떠 있다. 긴장과 집착속에 손을 떨구고 자유로와질 수 있는 시기. 모든 일은 한결같을 수 없듯이 우리의 삶에도 이렇게 재충전할 수 있는 여유는 다음일을 위한 활력소인 것 같다.

모든 것을 버릴 때 자유로와지는 것처럼 가벼운 짐하나 메고 여유로움 속으로 자! 이제 떠나자. 〈花〉